UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, STATION A, MONTRÉAL, QUÉBEC, H3C 3J7

TRANSCOM

TRANSCOM ET SON SUPERVISEUR:

DESCRIPTION TECHNIQUE

PAR

Claude Achim

Cette recherche a été subventionnée en partie par la compagnie Ford du Canada et le Centre de développement des transports.

Université de Montréal
Publication #25 - Centre de recherche sur les transports
Février 1976

TABLE DES MATIERES

INTRODUCTION 1

TRANSCOM EN RESUME 2

LES FICHIERS 10

- XRES 11
- NOUVO 13
- TABLE 14
- ODRES 16
- ODRST 17
- STATOR 19
- FLOW 21

ENTREE DES DONNEES 23

- BUILD NETWORK 24
- BUILD ACCESS TABLE 26
- BUILD OD MATRIX 28

LE SUPERVISEUR 29

- ANALYSE DES CARTES DE COMMANDE 30
- TYPES DE COMMANDES 32
- ORDRE DES COMMANDES 34
- LES PROGRAMMES DU SUPERVISEUR 35
- LES PROGRAMMES EXECUTANT LES COMMANDES . . . 39

CARTES DE CONTROLE 44

- BUILD NETWORK 47
- UPDATE NETWORK SCHEDULE 48
- DISPLAY NETWORK 49
- BUILD ACCESS TABLE 50
- DISPLAY ACCESS TABLE 52
- BUILD OD MATRIX 53

BUILD LINE FLOWS 55
DISPLAY FLOWS 56
DISPLAY PATHS 57

MODULE DE SERVICE 58

DESSINS SUR TRACEUR DE COURBES 59
TRI DU FICHIER ODRES 68

REFERENCES 77

INTRODUCTION

Le présent manuel s'adresse à l'analyste programmeur chargé de l'implantation et de l'utilisation de TRANSCOM. Celui-ci aura à définir des cartes de contrôle pour chacune des 9 tâches exécutables par TRANSCOM, allouer un support (espace sur disque) aux différents fichiers, superviser les travaux de l'analyste en transport (préparation de cartes de données et cartes de contrôle, soumission des programmes, gestion des fichiers, etc.) et maintenir à jour le système TRANSCOM. De plus, il pourra être appelé à concevoir de nouveaux programmes utilisant les fichiers générés par TRANSCOM, pour satisfaire à des besoins particuliers de l'analyste en transport.

NOTE: Nous supposons que le lecteur connaît le contenu du Guide de l'Utilisateur de TRANSCOM.

TRANSCOM EN RESUME

TRANSCOM se présente sous la forme de trois programmes fortran: PUBTRA, ACCES, CHARGE. Chacun contient un superviseur associé à un module de TRANSCOM et les ensembles de sous-programmes nécessaires à l'exécution des tâches propres à chaque sous-module, telles que décrites dans le Guide de l'Utilisateur. Cet ensemble est complété par des programmes sans superviseur, regroupés sous l'appellation Module de Service, qui permettent de réaliser des tâches particulières tel que tri des fichiers, dessins sur traceur de courbes, etc. Etant donné qu'ils font appel aux particuliarités de chaque installation, ils devront être définis au moment de l'implantation.

On peut considérer TRANSCOM comme un outil de création et de consultation des fichiers. Chacun de ces fichiers constitue une étape dans le processus de simulation du système de transport en commun. Les relations entre ces fichiers sont définies dans l'introduction du Guide de l'Utilisateur et représentées de façon schématique par les figures 1, 2, 3 et 4.

Un premier module, PUBTRAN, permet, à partir de données codifiées par l'analyste en transport, d'obtenir une image structurée du réseau de transport en commun. Cette image constitue le fichier de base de TRANSCOM, XRES.

PUBTRAN contient trois sous-modules:

BUILD NETWORK: création d'un fichier XRES

UPDATE NETWORK SCHEDULE: modification des temps d'attente ou de parcours à l'intérieur du fichier XRES:

on obtient alors un fichier nommé NOUVO qui pourra remplacer XRES ou devenir un deuxième XRES représentant une alternative au réseau à l'étude.

DISPLAY NETWORK: affichage structuré du contenu d'un fichier XRES

Les prochaines étapes sont réalisées par le module ACCES.

ACCES contient aussi trois sous-modules:

BUILD ACCESS TABLE: construction du fichier TABLE à partir de XRES et des coordonnées géocodées des centroides des zones origine-destination. TABLE contient une liste des noeuds du réseau accessibles à partir de chaque centroide.

DISPLAY ACCESS TABLE: consultation de la table d'accès.

BUILD OD MATRIX: construction du fichier ODRES. ODRES est obtenu à partir de XRES, TABLE, la matrice O-D zone à zone et de certains paramètres d'accès. Il contient une matrice O-D définie sur le réseau (i.e. noeud à noeud) obtenue par projection des centroides sur les noeuds du réseau.

On obtient le fichier ODRST en appliquant un tri du module de service (on utilisera avec avantage le tri du système d'exploitation de l'ordinateur local) sur le fichier ODRES. On peut alors éliminer le fichier ODRES devenu inutile, ODRST contenant alors la matrice O-D réseau triée.

Le module CHEMIN dont la tâche est de calculer les arbres des chemins optimaux à partir de chaque noeud du réseau et de les conserver sur le fichier TREES a été intégré au module CHARGE. Cependant, la version actuelle ne permet pas de conserver sur le fichier TREES les chemins calculés dans CHARGE.

Le module CHARGE permet de réaliser l'affectation des voyageurs sur le réseau de transport en commun et de constituer une banque d'informations statistiques sur chacun des noeuds du réseau considéré comme origine de déplacements.

Les trois sous-modules composant CHARGE sont:

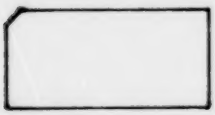
BUILD LINE FLOWS: création des fichiers FLOW et STATOR à partir de XRES, ODRST et d'un choix d'algorithme de cheminement.

DISPLAY FLOWS: consultation du fichier FLOW

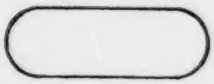
DISPLAY PATHS: impression des chemins utilisés pour réaliser l'affectation sur le réseau.

On doit noter qu'une simulation complète pourra nécessiter la création de plusieurs fichiers de chaque type dont la combinaison réflétera diverses hypothèses de comportement des usagers du transport en commun (perception des temps d'attente, politiques d'accès, choix d'itinéraire, etc.)

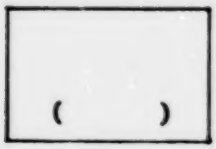
Les figures 1, 2, 3 et 4 résument l'utilisation de TRANSCOM et les relations entre les fichiers. Les différents symboles utilisés ont les significations suivantes:



fichier fourni par l'utilisateur généralement constitué de cartes de données sur fichier input



fichier généré par TRANSCOM et conservé sur disque



sous-module de TRANSCOM

on trouve entre parenthèses le nom de la sous-routine constituant le superviseur du sous-module

sortie sur imprimante

PUBTRAN

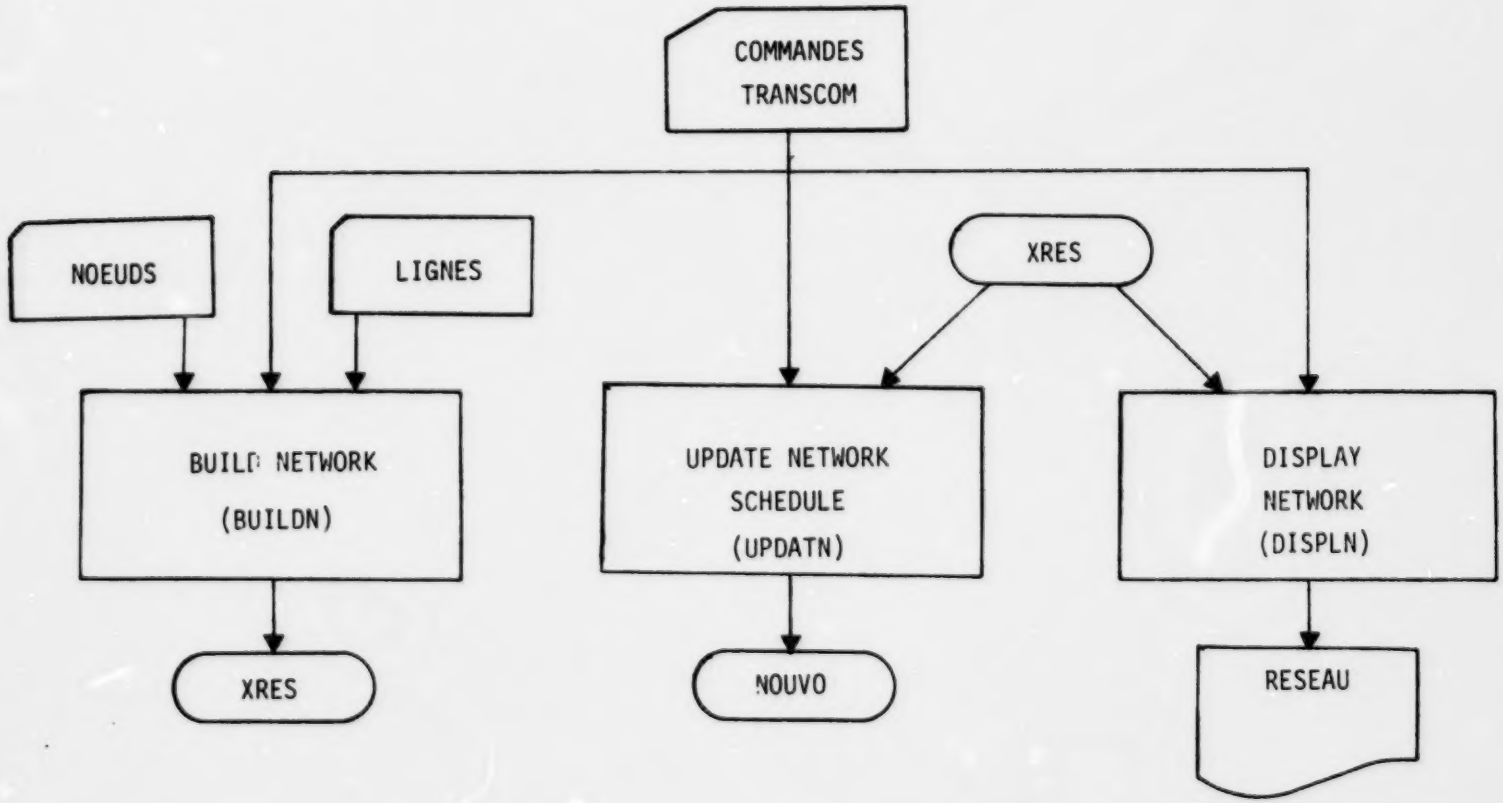


Figure 1

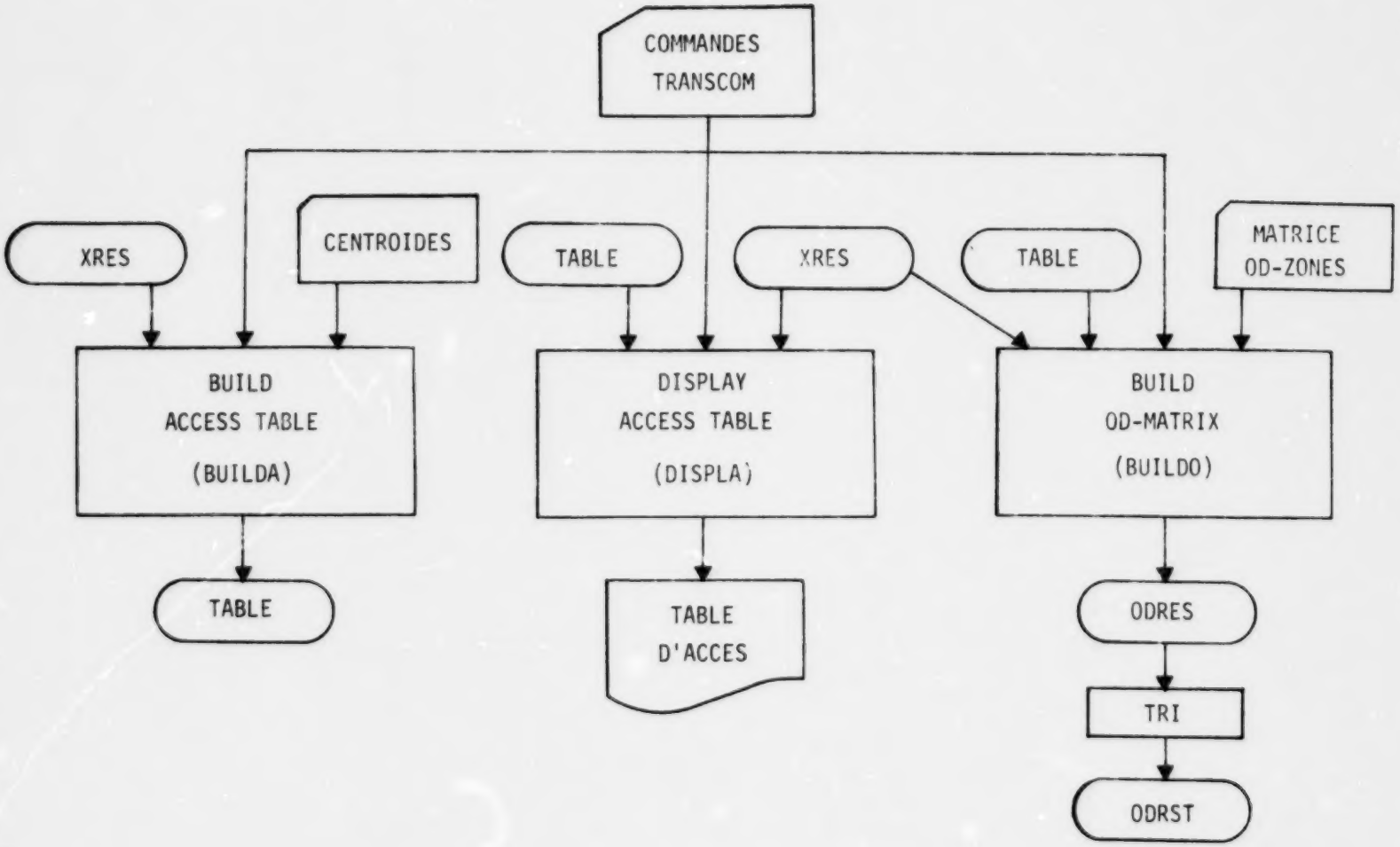

COMMANDES
TRANSCOM
XRES
CENTROIDES
TABLE
XRES
TABLE
MATRICE
OD-ZONES
BUILD
ACCESS TABLE
(BUILDA)
DISPLAY
ACCESS TABLE
(DISPLA)
BUILD
OD-MATRIX
(BUILDO)
TABLE
TABLE
D'ACCES
ODRES
TRI
ODRST

Figure 2

CHARGE

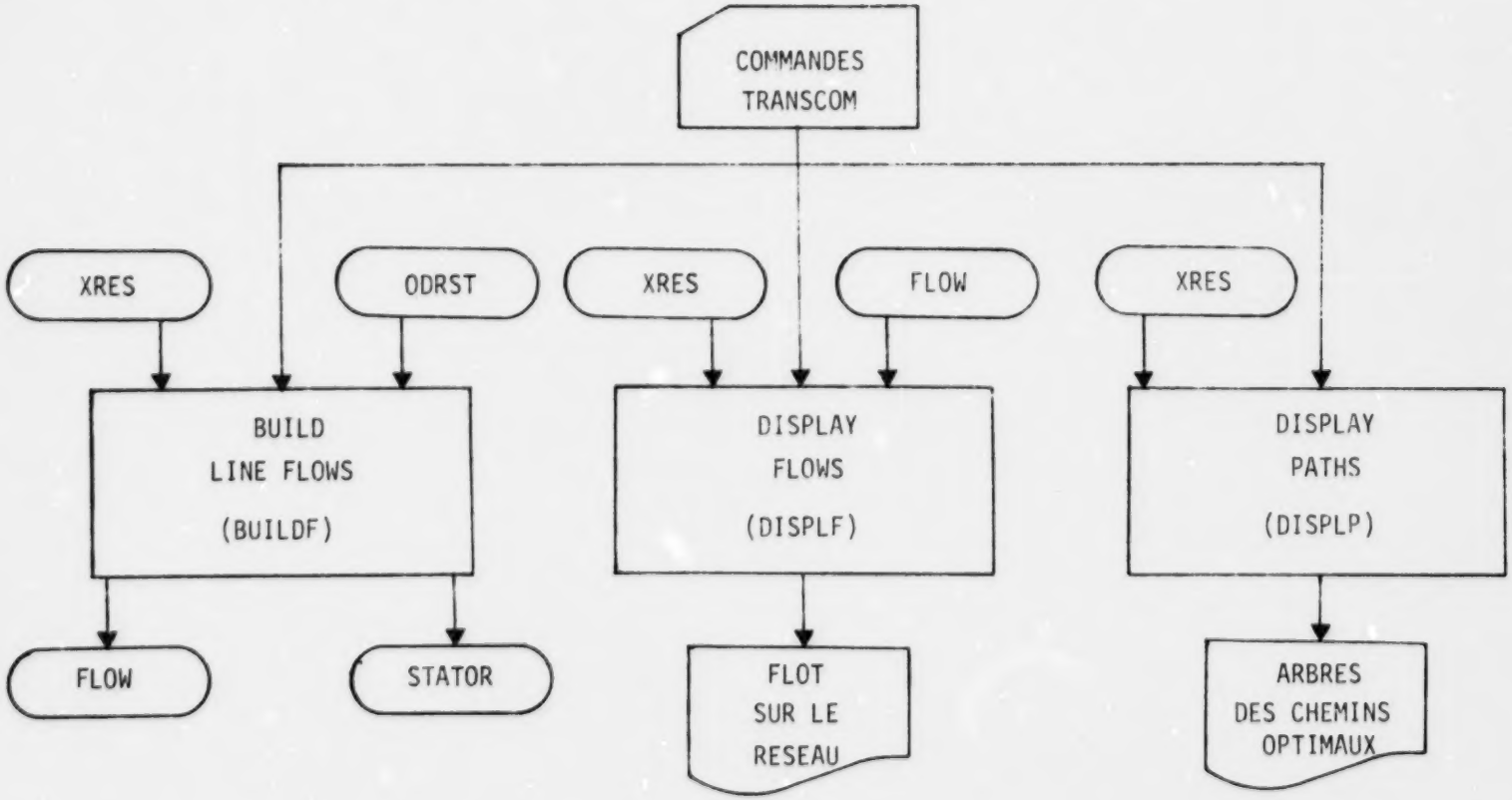


Figure 3

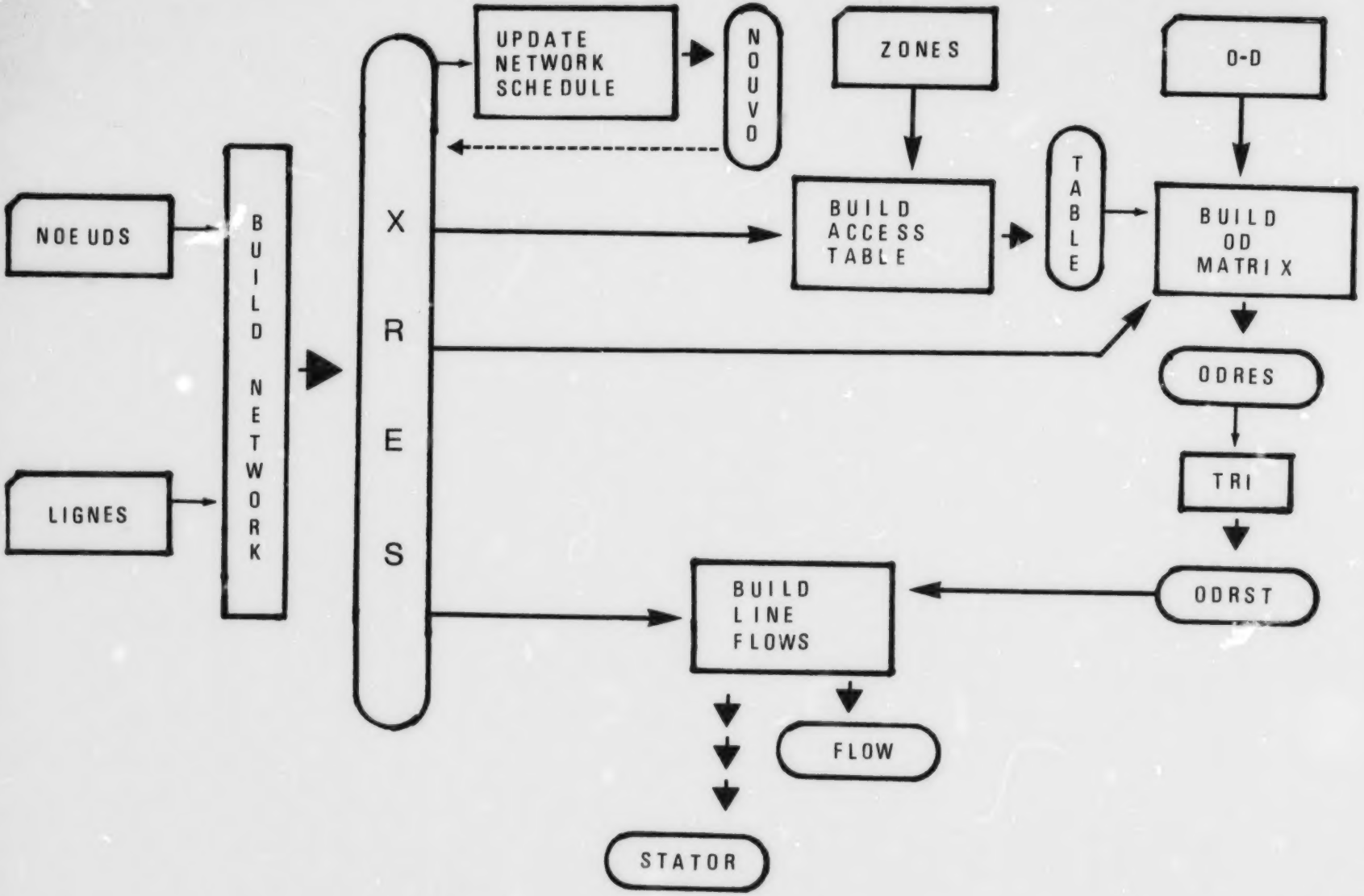

UPDATE NETWORK SCHEDULE
NOUVO
ZONES
O-D
NOEUDS
BUILD NETWORK
XRES
BUILD ACCESS TABLE
TABLE
BUILD OD MATRIX
ODRES
TRI
ODRST
LIGNES
BUILD LINE FLOWS
FLOW
STATOR

Figure 4

LES FICHIERS

TRANSCOM traite deux types de fichiers. Les "fichiers-utilisateur" sont constitués de cartes ou d'images de cartes et contiennent des données externes fournies par l'utilisateur. On retrouve dans cette catégorie les commandes au superviseur en langage TRANSCOM, les données sur les noeuds et les lignes du réseau, les coordonnées des centroides des zones et la matrice origine-destination. On traitera de ces fichiers dans la section Entrée des Données.

Les "fichiers-transcom" sont ceux créés par l'exécution d'un sous-module de TRANSCOM. Ces derniers possèdent un identificateur assigné (à l'aide de la commande KEEP du langage) par le superviseur et vérifié (commande USE) par celui-ci lors d'utilisation subséquente. Les fichiers XRES, NOUVO, TABLE, ODRES et ODRST sont binaires (WRITE binaire fortran) alors que FLOW et STATOR sont formatés (codés). Le premier mot ou record de ces fichiers est toujours son identificateur, un entier positif d'au plus dix chiffres décimaux. Ces fichiers seront généralement situés sur disque et on y aura accès par cartes de contrôle du système d'exploitation (voir section cartes de contrôle).

Nous donnons maintenant une description complète de chacun des"fichiers-transcom". Il est à noter pour les utilisateurs possédant une installation CDC avec système SCOPE que les noms XRES, NOUVO, TABLE, ODRES, ODRST, FLOW et STATOR sont aussi les noms SCOPE de ces fichiers. Pour les autres, on pourra toujours se référer au numéro d'unité qui est assigné à chaque fichier.

Sur C.D.C., les trois programmes commencent par la carte PROGRAM suivante:

```
      PROGRAM PUBTRA (INPUT,OUTPUT,XRES,NOUVO,TAPE8=NOUVO,TAPE9=XRES)

      PROGRAM ACCES (INPUT,OUTPUT,XRES,ZONES,ORIDES,TABLE,ODRES,
     1    TAPE1=ZONES,TAPE2=ORIDES,TAPE3=TABLE,TAPE4=ODRES,TAPE5=INPUT,
     2    TAPE6=OUTPUT,TAPE9=XRES)

      PROGRAM CHARGE (INPUT,OUTPUT,XRES,ODRST,TREES,STATOR,FLOW,
     1    TAPE3=STATOR,TAPE4=FLOW,TAPE5=TREES,TAPE6=ODRST,TAPE9=XRES)
```

1. XRES

Nom transcom[1]: NETWORK

Unité logique: TAPE9

Une variable du superviseur, TXRES, identifie l'unité logique. TXRES est initialisé dans les sous-routines BUILD, UPDATN, DISPLN, BUILDA, BUILDO, DISPLA, BUILDF, DISPLF, DISPLP par l'énoncé: TXRES = 9. Pour changer le numéro d'unité logique, il suffira de modifier cet énoncé dans les neuf sous-routines et de modifier en conséquence les trois cartes PROGRAM.

XRES est créé dans BUILDN par les énoncés suivants:

```
      REWIND TXRES
      WRITE (TXRES) IDNET
      WRITE (TXRES) (STBUF(J),  J = 1,19659)
```

---

1. Au niveau du langage, les fichiers sont désignés par leur nom transcom dans les commandes USE et KEEP.

XRES est lu dans les trois modules (PUBTRAN, ACCES, CHARGE), par un appel à la sous-routine INXRES:

```
REWIND IUNIT
READ (IUNIT) NAME
IF(EOF (IUNIT)) 10, 20
READ (IUNIT) (BUFF (J),  J = 1,19659)
IF(EOF (IUNIT)) 40, 50
```

Les appels à INXRES sont faits dans UPDATN, DISPLN, BUILDA, BUILDO, DISPLA, BUILDF, DISPLF, DISPLP.

XRES contient la structure représentant le réseau en mémoire. Cette structure est logée dans le COMMON/STRUC/ dont la définition est la suivante[1]:

```
COMMON/STRUC/  NNODE, NLINE, NOPREV, LIPREV, NSEG, NSEL, SCALE,
               METRO, MESTA,
               XNO(1000), YNO (1000), NOID (1000),
               LINID (150), VLIN (150), DESC (600), DMIL (150) LINTYP (150),
               FLIN (150), NOLIST (150), NONLIN (150),
               LISTN (2500), NDIR (2500), DA (2500), DR (2500),
               LISTL (2500), LPNT (2500)
```

La présense d'une commande au superviseur du type

```
USE NETWORK IDENT NUMBER nn
```

suppose qu'on a défini par carte contrôle un fichier XRES accessible pour lecture. Le superviseur vérifiera l'identificateur du fichier.

La commande au sous-module BUILD NETWORK

```
KEEP NETWORK IDENT NUMBER nn
```

indique que le programme génère un fichier XRES, lequel devra être conservé par carte contrôle.

---

2. NOUVO

Nom transcom : NETWORK

Unité logique: TAPE8

NOUVO est créé dans UPDATN par les énoncés suivants:

```
NXRES = 8
REWIND NXRES
WRITE (NXRES) NDNET
WRITE (NXRES) (STBUF (J),   J = 1,19659)
```

Comme pour TXRES, on peut changer la valeur de NXRES.

Pour ce qui est du contenu et de la lecture de NOUVO, on le considère comme un fichier de type XRES.

La commande au sous-module UPDATE NETWORK SCHEDULE

```
KEEP NETWORK IDENT NUMBER nn
```

indique que le programme génère un fichier NOUVO lequel devra être conservé par carte contrôle.

NOUVO est un fichier de type XRES, obtenu par modification de temps d'attente ou de parcours sur un fichier XRES initial. A la fin de l'exécution, on est en présence de deux fichiers dont les identificateurs peuvent être identiques ou différents. Si on ne désire pas conserver le fichier XRES initial, on devra l'éliminer de manière externe (i.e. il ne suffit pas de donner au superviseur un identificateur NOUVO qui soit le même que celui de XRES).

3. TABLE

Nom transcom : TABLE

Unité logique: TAPE3

TABLE est initialisé dans la sous-routine BUILDA:

```
TTABLE = 3
REWIND  TTABLE
WRITE (TTABLE) IDTAB
```

et les entrées de la table sont écrites dans la sous-routine ACCESS:

```
WRITE (3) ICENT, NODE, TEMPS, K, ITYPE
```

Le sous-module BUILD OD MATRIX utilise TABLE de la façon suivante:

1° sous-routine BUILDO, lecture de l'identificateur

```
TTABLE = 3
REWIND TTABLE
READ (TTABLE) KDTAB
IF (EOF(TTABLE)) 2220, 2230
```

2° sous-routines TAB1, TAB2, TAB3, TAB4, TAB5

```
5 READ (3) IC, IN, TA
  IF (EOF(3)) 50, 10
```

Le sous-module DISPLAY ACCESS TABLE utilise TABLE de la façon suivante:

1° sous-routine DISPLA, lecture de l'identificateur

```
TTABLE = 3
REWIND TTABLE
READ (TTABLE) NAME
IF (EOF(TTABLE)) 2020, 2030
```

2° sous-routine TABOUT

```
8 READ (3) ICENT, NODE,TEMPS, K, ITYPE
  IF(EOF(3)) 50,20
  .
  .
  BACKSPACE 3
```

TABLE contient la table d'accès complète des centroides des zones. Pour chaque centroide il y a autant d'entrées qu'il existe de combinaisons noeud accessible et mode d'accès. Les centroides apparaissent en ordre croissant d'identificateur dans cette table.

La commande au sous-module BUILD ACCESS TABLE

KEEP TABLE IDENT NUMBER nn

indique que le programme génère un fichier TABLE, lequel devra être conservé par carte contrôle.

Une commande au superviseur du module ACCES

USE TABLE IDENT NUMBER nn

suppose qu'on a défini par carte contrôle un fichier TABLE accessible pour lecture. La vérification de l'identificateur sera effectuée par le superviseur.

4. ODRES

Nom transcom : MATRIX

Unité logique: TAPE4

ODRES est initialisé dans la sous-routine BUILDO:

```
TODRS = 4
REWIND TODRS
WRITE (TODRS) IDODR
```

Les entrées de la matrice OD sont écrites dans la sous-routine TODRES par une série d'énoncés de la forme:

```
WRITE(4) IS, IT, XP
```

IS et IT représentent deux noeuds du réseau (numérotation interne) et XP représente le nombre de déplacements originant en IS et se destinant en IT. IS et IT sont des entiers alors que XP est réel. XP est toujours un multiple de 0.25. L'ensemble forme la matrice OD réseau obtenue à partir de la matrice OD zones.

La commande au sous-module BUILD OD MATRIX

```
KEEP MATRIX IDENT NUMBER nn
```

indique que le programme génère un fichier ODRES, lequel devra être conservé par carte contrôle.

5. ODRST

Nom transcom : MATRIX

Unité logique: TAPE6

On obtient ODRST de la façon suivante:

1° copie de l'identificateur de ODRES sur ODRST

2° tri en ordre croissant des origines (IS) de ODRES. Toutes les entrées IS, IT, XP triées sont envoyées sur ODRST. On aura une entrée par record (binaire) i.e. : WRITE (unité) IS, IT, XP

Pour plus de détails, voir la section Module de Service (TRI).

ODRST est utilisé en entrée par le sous-module BUILD LINE FLOWS:

1° sous-routine BUILDF

```
REWIND 6
READ (6) ID
IF (EOF(6)) 1211, 1212
```

2° sous-routine LOANET

```
READ (6) IC, JC, XKC
KC = INTEG * XKC

READ (6) IC, JC, XKC
IF (EOF(6)) 76, 77
77 KC = INTEG * XKC
```

Note: INTEG est initialisé à 4 dans BUILDF.

La commande au sous-module BUILD LINE FLOWS

USE MATRIX IDENT NUMBER nn

suppose qu'on a défini par carte contrôle un fichier ODRST accessible pour lecture. La vérification de l'identificateur sera effectuée par le superviseur.

6. STATOR

Nom transcom : STATOR

Unité logique: TAPE3

Au moment de l'affectation des usagers sur le réseau, le module CHARGE compile plusieurs statistiques associées aux noeuds du réseau considérés comme origine de déplacements. Toutes ces informations sont conservées sur le fichier STATOR.

STATOR est d'abord initialisé dans la sous-routine BUILDF:

```
      FAC = FAC/INTEG
      REWIND 3
      WRITE (3,7000) IDSTAT
      WRITE (3,7001) FAC
 7000 FORMAT (I10)
 7001 FORMAT (G20.12)
```

FAC est le facteur d'expansion appliqué à la matrice origine-destination (généralement obtenue par échantillonnage). INTEG est un multiplicateur (dont la valeur est 4) appliqué aux entrées de la matrice au moment du chargement, permettant d'obtenir des flots entiers.

Ensuite, pour chaque origine active du réseau, la sous-routine LOANET ajoute l'entrée suivante:

```
      WRITE (3,201) IBAS, FLOT, TMS, ANST, TATM, DM, DMS, PIED, DPIED
  201 FORMAT (I5, F8.0, 5F8.2, F8.0, F8.2)
```

IBAS : numéro (interne) du noeud origine

FLOT : nombre total de déplacements à partir de cette origine (devra être multiplié par FAC)

TMS : moyenne du temps total de voyage (minutes)

ANST : moyenne du nombre de sections de lignes empruntées

TATM : moyenne du temps total d'attente (minutes)

DM : moyenne de la distance à vol d'oiseau de section en section entre l'origine et la destination (milles)

PIED : nombre d'usagers marchant à pied sur le réseau (devra être multiplié par FAC)

DPIED: distance moyenne de marche pour ces usagers (milles)

STATOR a été prévu comme fichier d'entrée à des programmes du module de service non disponibles dans la version actuelle. Cependant son contenu est imprimé lors de l'exécution de BUILD LINE FLOWS.

La commande

KEEP STATOR INDENT NUMBER nn

indique que le programme génère un fichier STATOR, lequel devra être conservé par carte contrôle.

7. FLOW

Nom transcom : FLOW

Unité logique: TAPE4

FLOW contient deux types d'informations: le flot sur les segments de ligne et le nombre de personnes entrant, sortant et transférant aux différents noeuds du réseau. Son initialisation dans BUILDF est similaire à celle de STATOR:

```
      FAC = FAC/INTEG
      REWIND 4
      WRITE (4,7000) IDFLOW
      WRITE (4,7001) FAC
 7000 FORMAT (I10)
 7001 FORMAT (G20.12)
```

Le reste de son contenu est défini dans LOANET par:

```
      WRITE (4,230) (DY(I), DZ(I), I = 1, NSEG)
      WRITE (4,231) (NDPL(I), NDPLT(I), I = 1, NNODE)
  230 FORMAT (16F5.0)
  231 FORMAT (5(I10,I5))
```

DY et DZ, charge sur les segments du parcours aller et retour respectivement, doivent être multipliés par FAC. Leur relation avec la structure du fichier XRES est la même que celle des variables DA et DR (temps aller et temps retour sur les segments).

NDPL contient 10000 * nombre entrants + nombre sortants
on devra séparer ces deux quantités avant de multiplier par FAC

NDPLT indique le nombre de transferts effectués au noeud
(il faudra multiplier par FAC).

FLOW est lu par la sous-routine INFLOW appelée dans le sous-module DISPLAY FLOWS (sous-routine DISPLF):

```
          REWIND IUNIT
          READ (IUNIT, 100) NAME
          IF (EOF (IUNIT)) 10,20
          .
          .
          .
       30 READ (IUNIT, 110) FAC
          IF (EOF (IUNIT)) 10,40
       40 READ (IUNIT, 120) (DY(J), DZ(J), J = 1, NSEG)
          IF (EOF (IUNIT)) 10,50
       50 READ (IUNIT, 130) (NENTRE (J), NSORT(J), NTRANS(J), J = 1, NNODE)
          IF (EOF (IUNIT)) 10,60
      100 FORMAT (I10)
      110 FORMAT (G20.12)
      120 FORMAT (16F5.0)
      130 FORMAT (5(I6, I4, I5))
```

IUNIT est défini dans DISPLF par TFLOW = 4 .

La commande au sous-module BUILD LINE FLOWS

```
KEEP FLOW IDENT NUMBER nn
```

indique que le programme génère un fichier FLOW, lequel devra être conservé par carte contrôle.

La commande à DISPLAY FLOWS

```
USE FLOW IDENT NUMBER nn
```

suppose qu'on a défini par carte contrôle un fichier FLOW accessible pour lecture. La vérification de l'identificateur sera effectuée par le superviseur.

REMARQUE: Tous les fichiers de type transcom sont rebobinés au début de l'exécution mais non à la fin.

ENTREE DES DONNEES

Un bloc de commandes du langage TRANSCOM (voir Guide de l'Utilisateur) doit apparaître comme première série de données pour chacun des trois programmes, PUBTRAN, ACCES et CHARGE. Ce bloc identifie le sous-module qui sera exécuté et donne au superviseur de ce sous-module toutes les commandes nécessaires à la mise en place de cette exécution. Certains sous-modules de PUBTRAN et ACCES requièrent aussi certaines données supplémentaires. Celles-ci sont toujours sur cartes dans le cas de PUBTRAN et peuvent être sur cartes ou sur un autre fichier (ruban ou disque) dans le cas d'ACCES. Les données supplémentaires sur cartes (INPUT) sont placées immédiatement après la carte END du bloc de commande.

Nous décrivons ici le contenu et le format de ces données supplémentaires. Les formats de la version expérimentale de TRANSCOM présentés dans [2] et [3] diffèrent légèrement de ceux de la présente version et doivent être ignorés.

NOTE: Chaque exécution d'un programme de TRANSCOM ne traite qu'un seul bloc de commandes.

1. Sous-module BUILD NETWORK

Le fichier XRES est construit à partir des données résultant de la codification du réseau. On trouve d'abord la liste des noeuds du réseau et leurs coordonnées, suivie des cartes décrivant chacune des lignes.

Pour les noeuds on a 5 triplets (ID, X, Y) par carte où:

ID = identificateur du noeud
X = coordonnée sur l'axe des X
Y = coordonnée sur l'axe des Y

Le format de lecture est (5 (1X, I4, 2F5.0))

Un triplet contenant un identificateur de noeud négatif indique la fin de cette liste.

Un triplet contenant un identificateur nul (0 ou blanc) est ignoré.

Après la dernière carte de la liste des nceuds on trouve les cartes de description de lignes.

Pour chaque ligne, on a une première carte donnant les caractéristiques de la ligne et contenant un L en première colonne:

on y lit: NUMERO, NOM, LONGUEUR, INTERVALLE, VITESSE et TYPE de la ligne dans les champs suivants:

Colonne 1 :	L	(A1)
2-5 :	NUMERO	(I4)
6-20:	NOM	(3A4,A3)
21-25:	LONGUEUR (MILLES)	(F5.2)
26-30:	INTERVALLE (MINUTES)	(F5.1)
31-35:	VITESSE (MI/HRE)	(F5.1)
36-40:	TYPE	(I5)

Si TYPE est négatif, la ligne sera considérée comme fictive par les modules ACCES et CHARGE.

Cette première carte est suivie d'autant de cartes qu'il est nécessaire pour décrire le parcours de la ligne.

Une carte de parcours peut contenir 15 couples[1] (SENS,NOEUD) suivis d'un indicateur de continuation.

Le format de lecture est (15 (A1, I4), I5)

SENS prend la valeur: A si le noeud est sur le parcours aller seulement
R si le noeud est sur le parcours retour seulement
blanc si le noeud est parcouru dans les deux sens.

Une valeur négative en colonnes 76 à 80 indique que la description du parcours est continuée sur la prochaine carte.

On indique la fin des cartes décrivant les lignes par une carte blanche.

LIMITES: nombre de noeuds < 1000
nombre de lignes ≤ 150
nombre de segments ≤ 2500

---

1. Un couple contenant un identificateur de noeud nul (0 ou blanc) sera ignoré.
Un couple dont l'indicateur de sens prend une valeur autre que (A, R, blanc) sera aussi ignoré.

2. Sous-module BUILD ACCESS TABLE

Le "fichier-utilisateur" ZONES contient les coordonnées des centroides des zones associées à la matrice origine-destination.

Une commande au superviseur définit la localisation et le type de contenu de ce fichier:

ZONES ON INPUT

indique que le fichier est sur cartes après la carte END (unité logique TAPE5)

ZONES ON TAPE

indique que le fichier doit être lu sur l'unité logique TAPE1. Ce fichier devra alors être rendu accessible pour lecture par carte contrôle et sera rebobiné par le superviseur au début de l'exécution.

NOTE: ZONES est à la fois le nom transcom et le nom scope de ce fichier (version C.D.C. - SCOPE)

Le fichier ZONES peut être présenté sous deux formats différents, selon que la numérotation des centroides est explicite ou implicite.

1er cas, numérotation EXPLICITE:

Ce format est indiqué au superviseur par la commande:

ZONES ON [INPUT / TAPE] EXPLICIT IDENT

Le programme lit un triplet (NO, IX, IY) en format (3I5) par carte.

Les cartes doivent être placées en ordre croissant d'identificateur.

LIMITE: $1 \leq NO \leq 1200$

2^e cas, numérotation IMPLICITE:

Ce format sera utilisé si la particule "EXPLICIT IDENT" n'apparait pas sur la carte

ZONES ON [INPUT / TAPE]

Le programme lit alors 8 paires de coordonnées (IX, IY) par carte en format (8 (2I5)). L'ordre d'occurence tient lieu d'identificateur.

LIMITE: au plus 1200 paires de coordonnées.

Dans les deux cas, on indique la fin du fichier par une coordonnée IX dont la valeur est 99999.

Un centroide dont la coordonnée IX est nulle (0 ou blanc) sera ignoré. Cependant, dans le cas d'identification implicite, le compteur de centroides sera augmenté.

3. Sous-module BUILD OD MATRIX

Le "fichier-utilisateur" ORIDES contient la matrice origine-destination définie sur les centroides des zones. Une commande au superviseur définit la localisation de ce fichier:

OD ON INPUT

indique que le fichier est sur cartes à la suite de la carte END (unité logique TAPE5)

OD ON TAPE

indique que le fichier doit être lu sur l'unité logique TAPE2. Ce fichier devra alors être rendu accessible pour lecture par carte contrôle et sera rebobiné par le superviseur au début de l'exécution.

NOTE: OD est le nom transcom du fichier alors que ORIDES est son nom SCOPE (version C.D.C. - SCOPE)

ORIDES contient 6 triplets (O, D, N) par carte en format (18I4).

- O: numéro du centroide origine
- D: numéro du centroide destination
- N: nombre de déplacements de O à D.

Tout triplet dont l'origine est négative ou nulle sera ignoré. La lecture s'arrête sur une fin de fichier (END OF FILE).

LE SUPERVISEUR

Dans les programmes de TRANSCOM, la communication usager-programme est effectuée à l'aide d'un ensemble de sous-programmes appelé superviseur. Le langage défini dans le Guide de l'Utilisateur permet à l'usager de passer des commandes au superviseur. Le superviseur de module, après analyse de la carte de bloc, appelle le superviseur du sous-module correspondant à ce bloc de commandes. C'est au niveau du superviseur de sous-module que s'effectuent l'analyse et la validation des commandes, l'initialisation et la vérification des paramètres définis par l'utilisateur, le contrôle des identificateurs de fichiers de type transcom utilisés en entrée et l'attribution d'identificateurs aux fichiers transcom créés par l'exécution du programme. Ces unitialisations et vérifications étant terminées, le superviseur imprime un résumé des conditions d'exécution, après quoi, s'il n'y a pas d'erreur, il remet le contrôle à l'ensemble de sous-programmes réalisant la tâche requise.

1. Analyse des cartes de commande

Chaque commande est considérée par le superviseur comme une suite structurée de mots et de paramètres. Dans chaque module, le superviseur a accès à un dictionnaire contenant la définition (suite de caractère) de chacun des mots du langage propres à ce module. Le traitement de chaque carte est fait séquentiellement de la façon suivante:

1) lecture des 80 caractères de la carte
2) impression de cette carte
3) s'il s'agit d'une carte commentaire ou d'une carte blanche, lecture de la prochaine carte
4) identification de la commande et des paramètres, si la commande est inconnue ou s'il y a erreur, impression d'un message
5) s'il s'agit de la carte END, fin de l'analyse sinon retour à 1)

La première carte doit être une carte de bloc et la dernière une carte END. Généralement, les autres commandes peuvent apparaître dans un ordre quelconque. Des exceptions à cette règle seront signalées plus loin.

Le format des cartes est libre (les espaces entre les mots et à l'intérieur des mots sont ignorés) mais chaque commande doit occuper une seule carte. Les colonnes suivant le dernier caractère significatif d'une commande ne sont pas analysées et peuvent contenir des commentaires.

Les nombres définissant les paramètres numériques doivent respecter les règles suivantes:

- 1er caractère: + , - ou chiffre
- pas d'espace entre les caractères
- fin du nombre sur premier caractère non numérique sauf pour premier point décimal (.)
- le point décimal doit être précédé d'au moins un chiffre
  valides: 133 -133.3 0.77 +122.
  non valides: .32 +.32

2. Types de commandes

a) définition de paramètres

Certaines commandes permettent à l'usager de définir les paramètres externes nécessaires à l'exécution de la tâche. Ces paramètres sont validés et transmis aux sous-programmes par le superviseur. De plus, le superviseur vérifie que tous les paramètres obligatoires ont été définis.

Ex.: SCALE a UNITS = ONE MILE

b) impression de résultats

L'utilisateur peut commander l'impression de résultats durant l'exécution de la tâche. Ces demandes sont transmises comme des paramètres.

Ex.: LIST NODE CARDS

c) impression de fichiers

L'utilisateur peut demander d'imprimer de façon structurée le contenu de certaines parties de fichiers de transcom.

Ex.: DESCRIBE NODE n

d) prétraitement

Ces commandes permettent de modifier localement le contenu de certains fichiers, préalablement à l'exécution d'une tâche.

Ex.: LINE n DIVIDE WAITING TIME BY a

e) gestion de fichiers

Deux commandes, USE et KEEP permettent à l'utilisateur de définir les fichiers utilisés. Le superviseur peut alors vérifier les identificateurs

définis par les cartes USE et attribuer aux nouveaux fichiers les identificateurs définis par les cartes KEEP.

Dans certains cas, USE NETWORK et USE FLOW, le superviseur prend lui-même en charge la lecture du contenu des fichiers.

3. Ordre des commandes

Les commandes de prétraitement sont exécutées dans l'ordre d'apparition dans le bloc de commandes. De plus, ces commandes doivent apparaître après la carte USE NETWORK.

Les cartes mentionnant un identificateur de noeud ou de ligne du réseau font appel à la représentation interne du réseau. Elles doivent donc apparaître après la carte USE NETWORK qui commande l'entrée en mémoire du réseau.

4. Les programmes du superviseur

Nous donnons ici une liste des sous-programmes du superviseur de chaque module ainsi qu'un résumé de leur contenu.

a) sous-routines communes aux trois modules

GETNUM : lecture d'un nombre dans le vecteur KARTE

MATCH : transformation des caractères numériques en nombres

WORD : vérification de correspondance entre un mot du dictionnaire et une suite de caractères dans KARTE

NEXT : définit un pointeur au prochain caractère non blanc dans le vecteur KARTE

LECT : lecture et impression de la prochaine carte du bloc de commandes

INUM : traitement de la section IDENT NUMBER des cartes USE et KEEP

ERREUR : imprime un message d'erreur et donne à la variable EXEC la valeur .FALSE.

INXRES : lecture du fichier XRES

Tous ces sous-programmes apparaissent dans les trois modules et sont définis par des énoncés identiques. D'autres sous-programmes décrits plus bas apparaissent aussi dans les trois modules. Ils ont alors les mêmes fonctions mais certains énoncés diffèrent d'un module à l'autre.

b) Module Pubtran

PUBTRA : superviseur du module, analyse de la carte de bloc (programme principal)

BLOCK DATA : initialisation du dictionnaire

USE : analyse de la carte USE

KEEP : analyse de la carte KEEP

JOBSU : analyse de la carte JOB SUBMITTED BY

PRIN : analyse de la carte PRINT

BUILDN : superviseur du sous-module BUILD NETWORK

NUMBO : analyse de la carte NUMBER OF METRO

SCAL : analyse de la carte SCALE

ESTIM : analyse de la carte ESTIMATED NUMBER OF

LIST : analyse de la carte LIST (NODE, LINE) CARDS

COMP : analyse de la carte COMPARE LINE LENGTHS

UPDATN : superviseur du sous-module UPDATE NETWORK SCHEDULE

LINE : analyse de la carte LINE

INCRIS : analyse de la carte INCREASE WAITING TIME

DIVID : analyse de la carte DIVIDE WAITING TIME

DISPLN : superviseur du sous-module DISPLAY NETWORK

DESCR : analyse de la carte DESCRIBE

GRAP : analyse de la carte GRAPH NETWORK

NETU : analyse de la carte NETWORK SYSTEM UNITS

c) Module Accès

ACCES : superviseur du module, analyse de la carte de bloc (programme principal)

BLOCK DATA : initialisation du dictionnaire

USE : analyse de la carte USE

KEEP : analyse de la carte KEEP

JOBSU : analyse de la carte JOB SUBMITTED BY

BUILDA : superviseur du sous-module BUILD ACCESS TABLE

SCAL : analyse de la carte SCALE

AXS : analyse de la carte ACCESS

AXSDI : analyse de la carte ACCESS DISTANCE

AXSSP : analyse de la carte ACCESS SPEED

ZON : analyse de la carte ZONE

ZONON : analyse de la carte ZONES ON

ZONSY : analyse de la carte ZONE SYSTEM TO NETWORK SYSTEM

ZONTO : analyse de la carte définissant la transformation d'axes

SPE : analyse de la carte SPECIAL AREA

SPECIR : analyse de la carte décrivant une région spéciale circulaire

SPEREC : analyse de la carte décrivant une région spéciale rectangulaire

DIVID : analyse de la carte DIVIDE WAITING TIME

INCRIS : analyse de la carte INCREASE WAITING TIME

BUILDO : superviseur du sous-module BUILD OD MATRIX

REDU : analyse de la carte REDUCE ACCESS TABLE

SPC : analyse de la carte SPECIAL ACCESS ZONE

OD : analyse de la carte OD ON

DISPLA : superviseur du sous-module DISPLAY ACCESS TABLE

ZONED : analyse de la carte ZONE pour le bloc DISPLAY ACCESS TABLE

d) Module Charge

CHARGE : superviseur du module, analyse de la carte bloc (programme principal)

BLOCK DATA : initialisation du dictionnaire

USE : analyse de la carte USE

KEEP : analyse de la carte KEEP

JOBSU : analyse de la carte JOB SUBMITTED BY

DIVID : analyse de la carte DIVIDE WAITING TIME

INCRIS : analyse de la carte INCREASE WAITING TIME

LINE : analyse de la carte LINE

DESCR : analyse de la carte DESCRIBE

PRIN : analyse de la carte PRINT

BUILDF : superviseur du sous-module BUILD LINE FLOWS

COMPUT : analyse de la carte COMPUTE TREES

WALK : analyse de la carte WALKING DISTANCE

EXPA : analyse de la carte EXPANSION FACTOR

BUILDT : superviseur du sous-module BUILD MINPATH TREES (non-implanté)

DISPLF : superviseur du sous-module DISPLAY FLOWS

INFLOW : lecture du fichier FLOW

DISPLP : superviseur du sous-module DISPLAY PATHS

5. Les programmes exécutant les commandes

Nous donnons ici la liste des sous-programmes construits à partir des algorithmes décrits dans [2] et [3]. Sous la direction du superviseur, ils réalisent les tâches commandées par l'utilisateur. On trouvera dans [2] et [3] une liste de la version expérimentale de ces programmes. Ils ont subi certaines modifications dans la version actuelle.

a) Module Pubtran

NEXTA : localise le prochain noeud sur le parcours aller d'une ligne

NEXTR : localise le prochain noeud sur le parcours retour d'une ligne

STROUT : imprime la structure de représentation interne du réseau

NINT : retourne l'identificateur interne d'un noeud du réseau

LINT : retourne l'identificateur interne d'une ligne du réseau

STRUCT : fabrication de la structure de représentation interne à partir des données sur les noeuds et les lignes

LIDIVI : division du temps d'attente sur une ligne du réseau

LIADD : augmente le temps d'attente sur une ligne du réseau

ALADD : augmente les intervalles (temps d'attente) de toutes les lignes du réseau

ALDIVI : divise les intervalles de toutes les lignes du réseau

LITIME : modifie le temps de parcours d'un segment orienté sur une ligne du réseau

GETSEG : localise un segment orienté sur une ligne du réseau

DESSIN : dessin du réseau sur traceur de courbes (non-implanté, voir module de service)

NODESC : description de l'activité à un noeud du réseau

LIDESC : commande l'impression de la description d'une ligne du réseau

DIREC : détermine la direction d'une ligne du réseau (l'axe des Y définit la direction NORD-SUD)

TITRE : impression de l'entête de description de ligne

LIPRIN : impression de la description d'une ligne du réseau

SEGCOM : détermine les segments communs des lignes communes

b) Module Accès

NINT : retourne l'identificateur interne d'un noeud du réseau
(cette fonction est identique à NINT du module Pubtran)

ALDIVI : divise les intervalles de toutes les lignes du réseau

ALADD : augmente les intervalles de toutes les lignes du réseau

AC1 : détermine les rectangles de drainage

AC2 : entrée des coordonnées des centroides
transformation d'axes
détermination des lignes d'accès

AC3 : détermine les noeuds d'accès

IFINRS : détermine si le centroide appartient à une région spéciale

ACCESS : écriture d'une entrée sur le fichier TABLE

TAB1 : réduction de la table d'accès
politique = Plus Proche Noeud

TAB2 : réduction de la table d'accès
politique = Plus Proche Noeud Préférence Métro

TAB3 : réduction de la table d'accès
politique = Deux Plus Proches Noeuds

TAB4 : réduction de la table d'accès
politique = Plus Proche Bus et Plus Proche Métro

TAB5 : réduction de la table d'accès
politique = Plus Proche Noeud et Plus Proche Métro

TODRES : transformation de OD-centroides en OD-réseau non trié

BLOC : calcul du nombre de blocs associés à chaque origine

TABOUT : impression de la table d'accès d'un centroide

c) Module Charge

NINT : retourne l'identificateur interne d'un noeud du réseau

LINT : retourne l'identificateur interne d'une ligne du réseau

Les fonctions NINT et LINT sont identiques à celles du module Pubtran.

ALDIVI : divise les intervalles de toutes les lignes du réseau

ALADD : augmente les intervalles de toutes les lignes du réseau

LIDIVI : division du temps d'attente (intervalle) d'une ligne du réseau

LIADD : augmente le temps d'attente d'une ligne du réseau

STROUT : imprime la structure de représentation interne du réseau

La sous-routine STROUT est identique à celle de Pubtran. Cependant dans les résultats imprimés, les temps aller et retour sont remplacés par les flots aller et retour.

RUBAN : lecture des chemins sur un fichier

TRAMIN, PIVOTE : calcul des chemins à transfert minimum

LOANET : chargement du réseau à partir de la matrice OD et des calculs de chemins

LOASEC : chargement d'une section de ligne

CATM : calcul des chemins à temps minimum

ACCESS

RIGHTP

LEFTP

TRAITE

DELETE

INSER

MODIF

NEXTA : localise le prochain noeud sur le parcours aller d'une ligne

NEXTR : localise le prochain noeud sur le parcours retour d'une ligne

NEXTA et NEXTR sont identiques à celles du module Pubtran

NODESC : description de l'activité à un noeud du réseau

LIDESC : commande l'impression de la description d'une ligne du réseau

DIREC : détermine la direction d'une ligne du réseau

LIDESC et DIREC sont identiques à celles du module Pubtran

TITRE : impression de l'entête de description de ligne

LIPRIN : impression de la description d'une ligne du réseau

SEGCOM : détermine les segments communs et leur charge

ARBOUT : impression de l'arbre des chemins à partir d'une origine

CARTES DE CONTROLE

Nous ne pouvons décrire ici les cartes de contrôle particulières à chaque installation. Nous supposerons un système fictif par lequel nous définirons les cartes de contrôle nécessaires à chaque tâche. De là, chacun pourra définir des cartes de contrôle pour son installation locale.

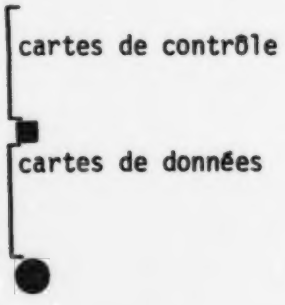
Quoiqu'il soit préférable d'utiliser des fichiers sur disque, supposons une installation où les fichiers sont sur rubans magnétiques.
La carte de contrôle:

MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, ECRITURE

Signifie que le programme fortran va utiliser un fichier sur ruban magnétique dont l'identificateur physique est 132. Ce fichier sera le fichier TAPE9 du programme fortran. Le programme écrira sur le ruban.

La carte: EXECUTER, PUBTRAN
signifie qu'on veut faire exécuter le programme fortran PUBTRAN.

Lors de l'exécution d'une tâche sous ce système hypothétique, le jeu de cartes soumis à la lectrice de cartes a la structure suivante:



Les rubans magnétiques disponibles pour cette simulation sont affectés de la façon suivante:

Fichiers TRANSCOM:

Numéro du ruban	fichier
132	XRES
160	XRES (modifié)
164	TABLE
167	ODRES
168	ODRST
201	STATOR
202	FLOW

Fichiers Utilisateurs:

050	ZONES
060	ORIDES

On trouvera à la figure 5 le schéma logique de la simulation.

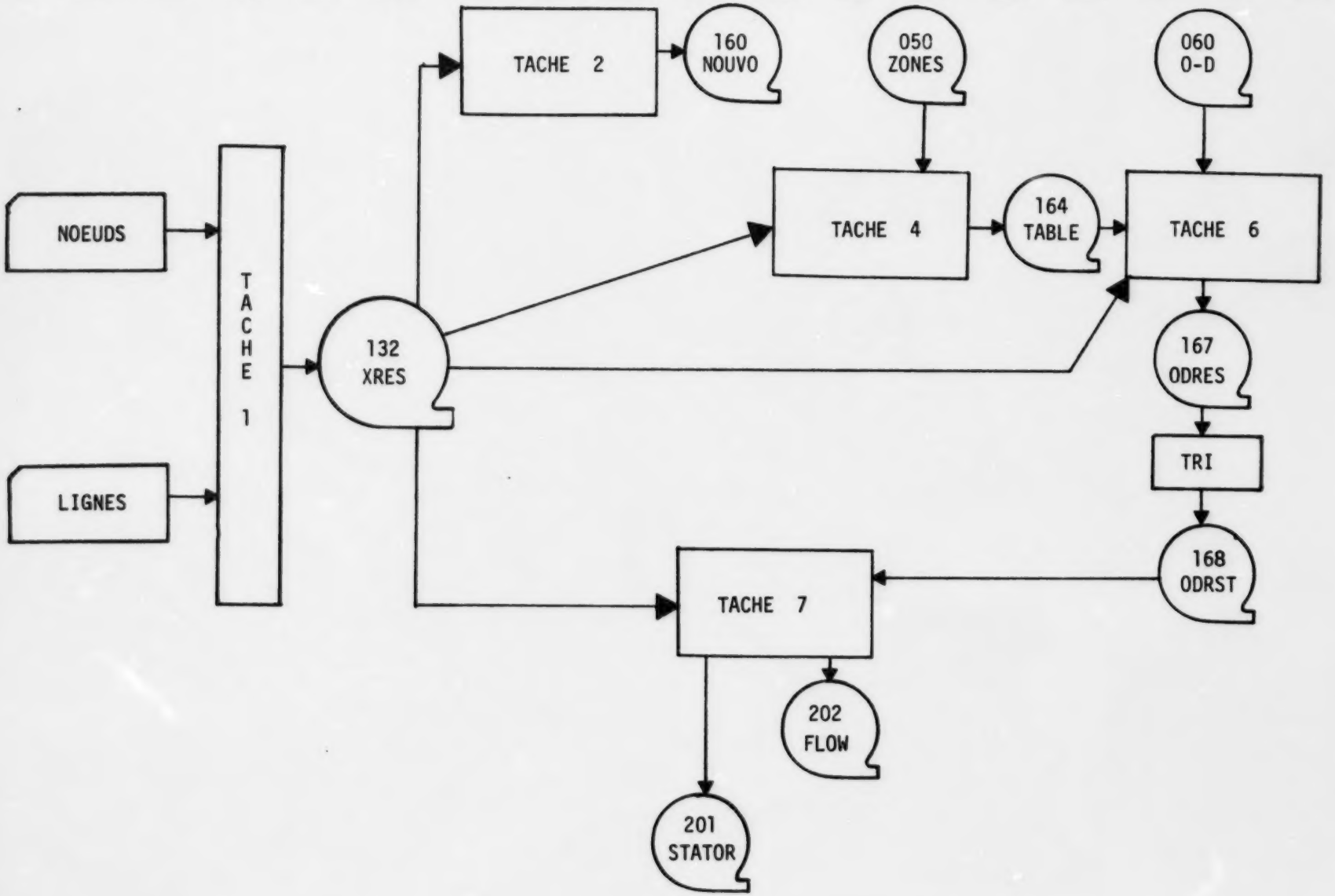

TACHE 2
160
NOUVO
050
ZONES
060
O-D
NOEUDS
TACHE 1
TACHE 4
164
TABLE
TACHE 6
132
XRES
167
ODRES
TRI
LIGNES
168
ODRST
TACHE 7
202
FLOW
201
STATOR

Figure 5

1. TACHE BUILD NETWORK

```
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, ECRITURE.
EXECUTER, PUBTRAN.
■

BUILD NETWORK
  :
  :
KEEP NETWORK IDENT NUMBER 1000
GO
END
  :    cartes de NOEUDS
-1
  :    cartes de LIGNES
  :
       carte blanche
●
```

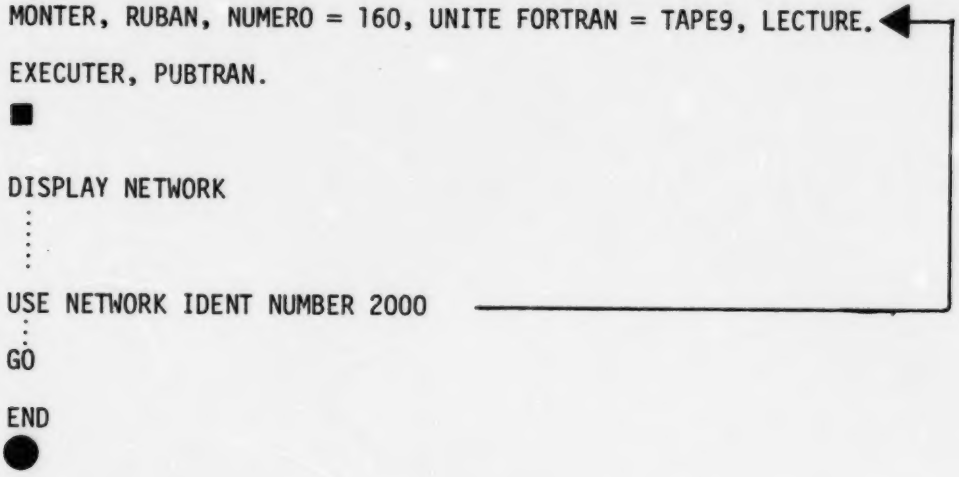
La structure interne du réseau sera conservée sur le ruban numéro 132.

2. TACHE UPDATE NETWORK SCHEDULE

```
MONTER, RUBAN, NUMERO = 160, UNITE FORTRAN = TAPE8, ECRITURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, PUBTRAN.
■

UPDATE NETWORK SCHEDULE
  ⋮
USE NETWORK IDENT NUMBER 1000
  ⋮
KEEP NETWORK IDENT NUMBER 2000
GO
END
●
```

La structure interne du réseau modifié est conservée sur le ruban numéro 160. La modification est faite à partir de l'ancien réseau (ruban 132).

3. TACHE DISPLAY NETWORK



On imprime des parties du réseau modifié. Le fichier transcom NETWORK dont l'identification est 2000 se trouve sur le ruban numéro 160.

4. TACHE BUILD ACCESS TABLE

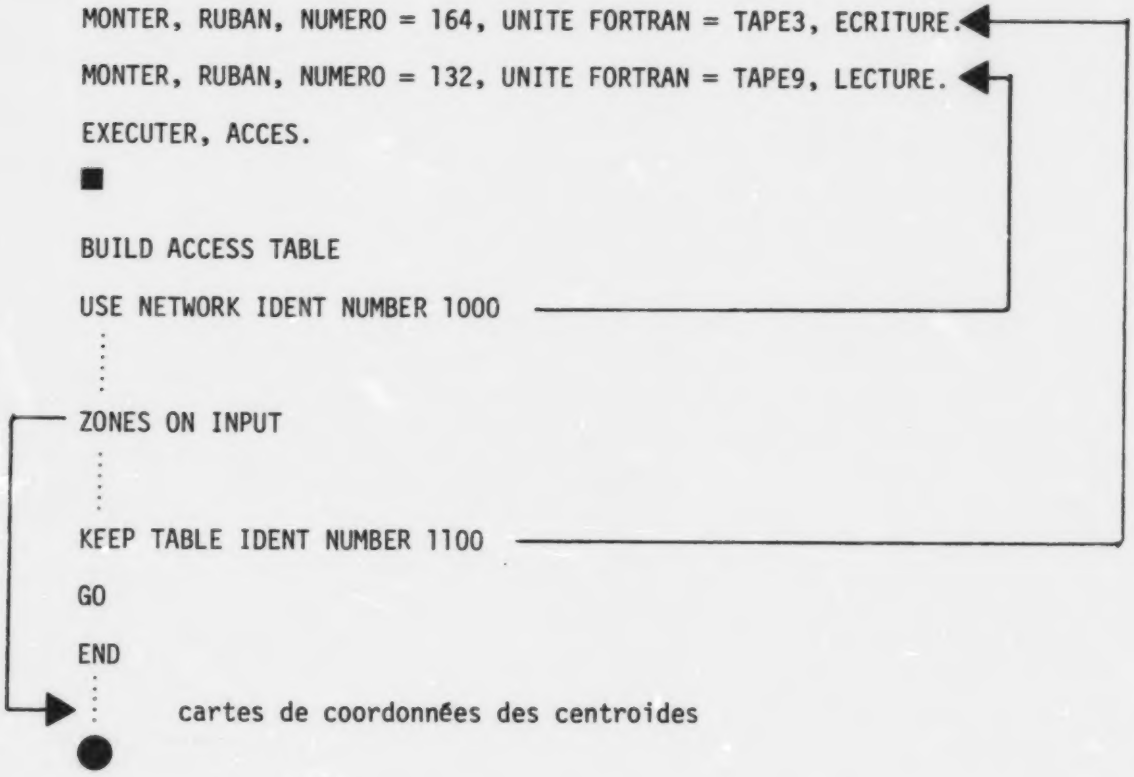
1er cas: les coordonnées des centroides sont sur un ruban magnétique dont le numéro est 050

```
MONTER, RUBAN, NUMERO = 164, UNITE FORTRAN = TAPE3, ECRITURE.
MONTER, RUBAN, NUMERO = 050, UNITE FORTRAN = TAPE1, LECTURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, ACCES.
■

BUILD ACCESS TABLE
USE NETWORK IDENT NUMBER 1000
  :
ZONES ON TAPE
  :
KEEP TABLE IDENT NUMBER 1100
GO
END
●
```

A partir du fichier NETWORK numéro 1000 (ruban 132) et des coordonnées des centroides sur le ruban 050, on crée une table d'accès. Le fichier TABLE reçoit l'identificateur 1100 et est conservé sur le ruban 164.

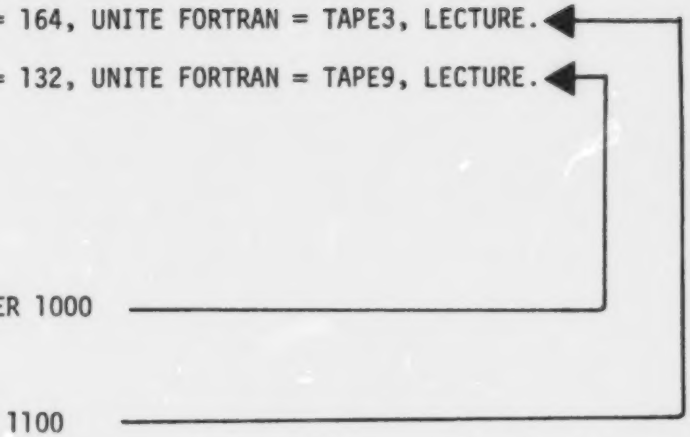
2^{e} cas: les coordonnées des centroides sont sur cartes perforées.

Pour réaliser une tâche equivalente au premier cas on aura:



5. TACHE DISPLAY ACCESS TABLE

```
MONTER, RUBAN, NUMERO = 164, UNITE FORTRAN = TAPE3, LECTURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, ACCES.
■

DISPLAY ACCESS TABLE
USE NETWORK IDENT NUMBER 1000
  :
USE TABLE IDENT NUMBER 1100
  :
GO
END
●
```



On imprime des sections du fichier TABLE numéro 1100 (ruban 164).

Cette table avait été créée pour le réseau NETWORK numéro 1000 (ruban 132).

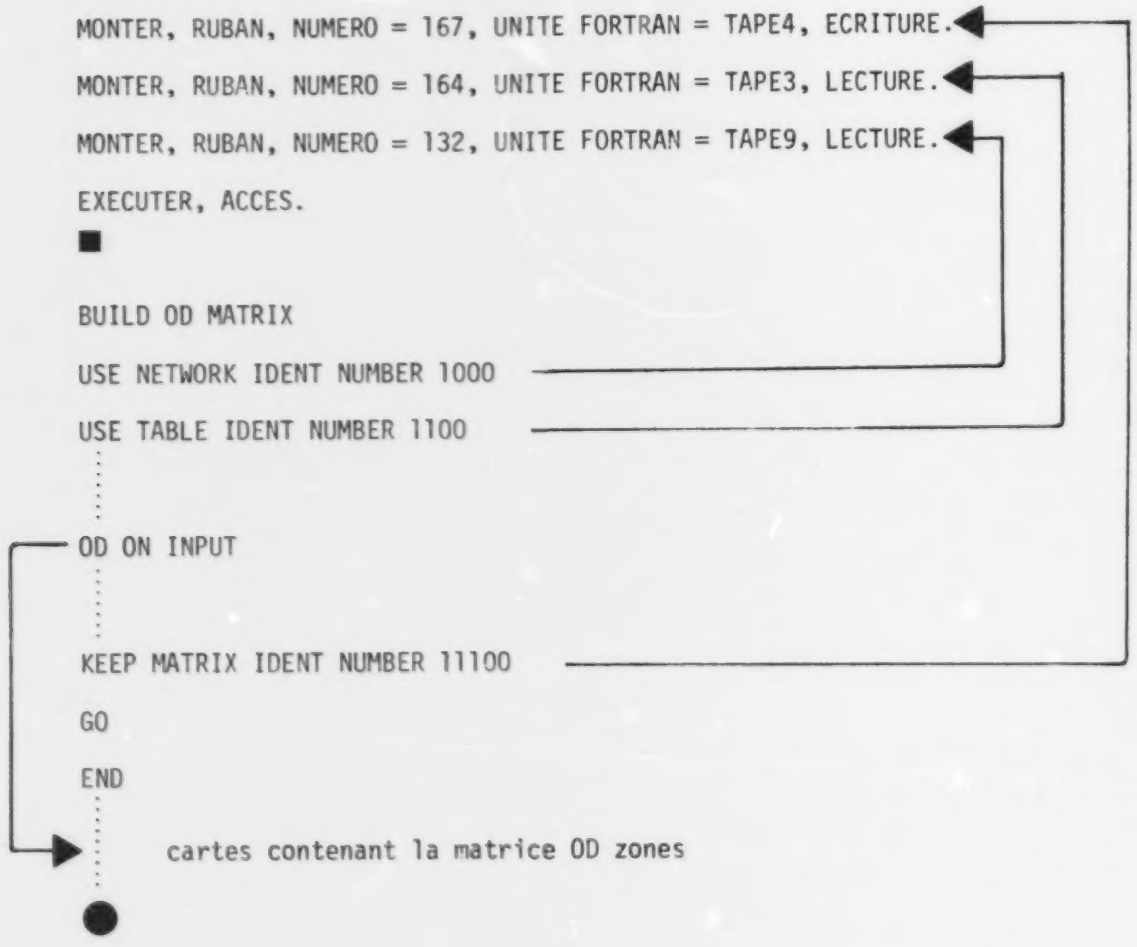
6. TACHE BUILD OD MATRIX

1er cas, matrice O-D zones sur ruban magnétique numéro 060.

```
MONTER, RUBAN, NUMERO = 167, UNITE FORTRAN = TAPE4, ECRITURE.
MONTER, RUBAN, NUMERO = 060, UNITE FORTRAN = TAPE2, LECTURE.
MONTER, RUBAN, NUMERO = 164, UNITE FORTRAN = TAPE3, LECTURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, ACCES.
■

BUILD OD MATRIX
USE NETWORK IDENT NUMBER 1000
USE TABLE IDENT NUMBER 1100
  :
  :
OD ON TAPE
  :
  :
KEEP MATRIX IDENT NUMBER 11100
GO
END
●
```

Le fichier MATRIX numéro 11100 créé par cette tâche sera écrit sur le ruban numéro 167.

2e cas, matrice O-D zones sur cartes perforées. Pour réaliser une tâche equivalente au premier cas on aura:



7. TACHE BUILD LINE FLOWS

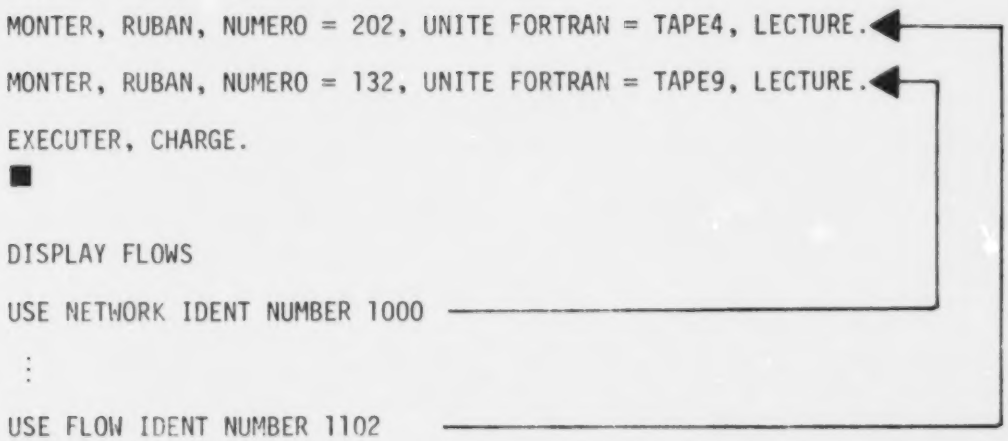
Préalablement à cette tâche, on doit trier la matrice OD réseau. Nous supposerons donc qu'on a appliqué un tri sur le ruban 167. Ce tri conserve le premier record du fichier (identificateur transcom). Le fichier trié est maintenant sur le ruban 168.

```
MONTER, RUBAN, NUMERO = 202, UNITE FORTRAN = TAPE4, ECRITURE.
MONTER, RUBAN, NUMERO = 201, UNITE FORTRAN = TAPE3, ECRITURE.
MONTER, RUBAN, NUMERO = 168, UNITE FORTRAN = TAPE6, LECTURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, CHARGE.
■

BUILD LINE FLOWS
USE NETWORK IDENT NUMBER 1000
  :
USE MATRIX IDENT NUMBER 11100
  :
KEEP STATOR IDENT NUMBER 1101
KEEP FLOW IDENT NUMBER 1102
GO
END
●
```

Le fichier FLOW numéro 1102 sera sur le ruban numéro 202 et le fichier STATOR numéro 1101 sera sur le ruban numéro 201.

8. TACHE DISPLAY FLOWS

```
MONTER, RUBAN, NUMERO = 202, UNITE FORTRAN = TAPE4, LECTURE.
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, CHARGE.
■

DISPLAY FLOWS
USE NETWORK IDENT NUMBER 1000
 :
USE FLOW IDENT NUMBER 1102
 :
GO
END
●
```



Les résultats du chargement seront imprimés à partir du fichier FLOW numéro 1102 (ruban 202) et du réseau correspondant (NETWORK 1000).

9. TACHE DISPLAY PATHS

```
MONTER, RUBAN, NUMERO = 132, UNITE FORTRAN = TAPE9, LECTURE.
EXECUTER, CHARGE.
■

DISPLAY PATHS
USE NETWORK IDENT NUMBER 1000
  :
  :
GO
END
●
```


A partir du fichier NETWORK 1000, des chemins seront calculés et imprimés sur papier.

Remarques:

1- DISPLAY PATHS est un sous-module de CHARGE parce qu'il utilise les algorithmes de calcul de chemins de ce module. Cependant, on peut l'utiliser à toute étape du processus de simulation. Par exemple, on peut faire imprimer des chemins pour vérifier la codification du réseau.

2- Pour chaque arbre commandé, le programme imprime autant de lignes sur papier, qu'il y a de noeuds dans le réseau. On devra donc utiliser cet outil parcimonieusement.

MODULE DE SERVICE

Nous appelons Module de Service, un ensemble de programmes utilitaires qui ne font pas partie comme tel du processus de simulation. Ces programmes, sans superviseur, utilisent les fichiers générés par TRANSCOM, pour répondre à des besoins spécifiques. Certains de ces programmes sont disponibles alors que d'autres font encore l'objet de recherches. L'utilisateur lui-même pourra ajouter à ce module des programmes associant des fichiers de TRANSCOM à des données locales.

Les programmes du Module de Service ont une caractéristique commune: ils doivent être adaptés à l'ordinateur utilisé et aux ressources disponibles localement. Nous donnerons ici une description des programmes DESRES et TRIODR qui permettent de réaliser des dessins de réseaux ou d'arbres (DESRES) et de trier en mémoire centrale le fichier ODRES (TRIODR).

1. Dessins sur traceur de courbes

La représentation graphique du réseau analytique s'avère un outil très important pour l'analyse des résultats d'une simulation ainsi que pour la vérification de la codification du réseau. On peut en effet inscrire directement sur les sections de lignes les flots de voyageurs obtenus par simulation. On peut alors comparer ces flots entre eux ou les comparer aux comptes réels. De même on peut procéder à des vérifications visuelles du réseau en cherchant des inconsistences (intersection de lignes sans noeud de transfert par exemple). Si le dessin est fait à l'échelle de la carte ayant servi à la codification, on pourra alors, par superposition, relever des anomalies de codification.

De même, le dessin d'un arbre des chemins les plus courts à partir d'une origine facilite l'analyse des temps de parcours et la calibration du réseau.

Le programme DESRES permet de réaliser de tels dessins sur le traceur de courbes CALCOMP de l'Université de Montréal. Ce programme fortran a été conçu pour l'ordinateur CDC et utilise les sous-routines de traçage de courbes disponibles à notre centre. La description que nous en donnerons ici fournira les informations nécessaires à l'écriture d'un programme semblable utilisant des ressources différentes.

Le programme DESRES réalise un dessin en quatre étapes:

1- lecture du fichier XRES et vérification de son identificateur

2- transformation des coordonnées des noeuds du réseau

3- traçage des noeuds et de leur identificateur

4- traçage des liens (du réseau ou de l'arbre)

Les données requises sont les suivantes:

1ère carte: identificateur de XRES en format I10

2e carte : XO, YO, ALFA en format 3F10.5

3e carte : FC en format F10.5

4e carte : INDIC, NOEUD en format 2I5

La transformation permet de passer du système d'axes de la codification au système d'axes du traceur. On applique une translation (XO, YO) et une rotation d'ALFA radians. (XO, YO) représente les coordonnées dans le système du réseau de l'origine (0, 0) du système d'axes du traceur. ALFA est l'angle mesuré en radians entre les deux systèmes d'axes (axes des X du réseau vers axe des X du traceur). FC est un facteur de conversion qui permet d'exprimer un pouce sur le dessin en nombre d'unités du système réseau.

La dernière carte de données indique le type de dessin désiré:

INDIC ≤ 0 : dessin du réseau au complet

INDIC = 1 : dessin d'un arbre d'origine NOEUD
algorithme TRAMIN

INDIC = 2 : dessin d'un arbre d'origine NOEUD
algorithme CTM

Nous présentons ici le programme DESRES associé à la version académique de TRANSCOM. Cette version est limitée à 200 noeuds, 50 lignes et 500 segments. Les principales variables de la sous-routine DESSIN sont les suivantes:

- structure du réseau:

XNO (I) : coordonnée X du noeud I

YNO (I) : coordonnée Y du noeud I

NOID (I) : identificateur du noeud I

NOLIST (J) : pointeur au premier noeud dans la liste des noeuds pour la ligne J

NONLIN (J) : pointeur au dernier noeud dans la liste des noeuds pour la ligne J

LISTN (K) : Kième noeud dans la liste des noeuds

NDIR (K) : sens de parcours du noeud LISTN (K)

- arbre des chemins depuis une origine

TIAC (I) : temps de voyage de l'origine au noeud I en minutes

NOAC (I) : dernier point de transfert pour atteindre le noeud I

LIAC (I) : ligne conduisant au noeud I

Les sous-programmes de traçage appelés par DESSIN sont:

PLOT : déplacer la plume

SYMBOL : trace un symbole ou un texte

NUMBER : trace un nombre

ENDPLT : termine le traçage

Les sous-programmes CATM et TRAMIN appelés par DESSIN pour le calcul des chemins sont ceux du module CHARGE. Leur rôle consiste à définir, pour une origine donnée, le contenu des vecteurs TIAC, NOAC et LIAC.

```
      PROGRAM DESRES(INPUT,OUTPUT,XRES,TAPE9=XRES)
      COMMON /STRUC/ BUF(4160)
      EQUIVALENCE (NAME,BUF(1))
C
C           CARTES DE DONNEES .........
C           1)  IDENTIFICATEUR XRES  (I10)
C           2)  XO,YO,ALFA   (3F10.5)
C           3)  FC  (F10.5)
C           4)  INDIC,NOEUD   (2I5)
C
C  IDENTIFICATEUR DU FICHIER
      READ 100,IDNET
      READ (9) BUF(1)
      IF(EOF(9)) 10,5
C  ENTREE DU RESEAU
    5 READ (9) (BUF(J),J=2,4160)
      IF(EOF(9)) 10,20
   10 PRINT 1000
      STOP
   20 IF(NAME.EQ.IDNET) GO TO 40
      PRINT 1001,NAME
      STOP
C           TRANSFORMATION DES COORDONNEES DES NOEUDS
C           EN POUCES DANS LE SYSTEME D AXES DU TRACEUR
C
C  LIMITES SUR L AXE Y DU TRACEUR ....
C  SI  Y.LT.YMIN  ALORS  Y=YMIN  (POUCES)
C  SI  Y.GT.YMAX  ALORS  Y=YMAX  (POUCES)
   40 YMIN=0.0
      YMAX=10.0
      CALL TRANFO(YMIN,YMAX)
C
C           LIRE LE TYPE DE DESSIN DESIRE
C
C  INDIC = 0 ... TOUTES LES LIGNES
C        = 1 ... ARBRE TRAMIN
C        = 2 ... ARBRE CTM
      READ 101,INDIC,NOEUD
C
C  REALISER LE DESSIN
      CALL DESSIN(INDIC,NOEUD)
      STOP
  100 FORMAT(I10)
  101 FORMAT(2I5)
 1000 FORMAT(///,18H FICHIER XRES VIDE )
 1001 FORMAT(///,21H IDENTIFICATEUR XRES  ,I10)
      END
```

```
      SUBROUTINE TRANFO(YMIN,YMAX)
      COMMON /STRUC/ NAME, NNODE,NLINE,NOPREV,LIPREV,NSEG,NSEL,SCALE,M,
     Q MESTA,XNO(200),YNO(200),NOID(200),
     Q LINID(50),VLIN(50),DESC(200),DMIL(50),LINTYP(50),
     Q FLIN(50),NOLTST(50),NONLIN(50),
     Q LISTN(500),NOIR(500),DA(500),DR(500),
     Q LISTL(500),LPNT(500)
C  TRANSLATION   (X0,Y0)   UNITES RESEAU
C  ROTATION      ALFA      RADIANS
      READ 100,X0,Y0,ALFA
C  FACTEUR DE CONVERSION :  FC UNITES RESEAU PAR POUCE DE DESSIN
      READ 100,FC
      DO 10 J=1,NNODE
      XP=(XNO(J)-X0)*COS(ALFA)+(YNO(J)-Y0)*SIN(ALFA)
      YP=(YNO(J)-Y0)*COS(ALFA)-(XNO(J)-X0)*SIN(ALFA)
C
C           TRANSFORMATION EN POUCES  (PRECISION = MILLIEME)
      IXP=(XP/FC)*1000.0
      IYP=(YP/FC)*1000.0
      XNO(J)=IXP/1000.0
      YNO(J)=IYP/1000.0
      IF(YNO(J).LT.YMIN) YNO(J)=YMIN
      IF(YNO(J).GT.YMAX) YNO(J)=YMAX
   10 CONTINUE
      RETURN
  100 FORMAT(3F10.5)
      END
```

```
      SUBROUTINE DESSIN(INDIC,NOEUD)
      COMMON /STRUC/ ST
      COMMON /STRUC/ NNODE,NLINE,NOPREV,LIPREV,NSEG,NSEL
      COMMON /STRUC/ SCALE,METRO,MESTA
      COMMON /STRUC/ XNO(200),YNO(200),NOID(200)
      COMMON /STRUC/ LINID(50),VLIN(50),DESC(200),DMIL(50),LINTYP(50)
      COMMON /STRUC/ FLIN(50),NOLIST(50),NONLIN(50)
      COMMON /STRUC/ LISTN(500),NDIR(500),DA(500),DR(500)
      COMMON /STRUC/ LISTL(500),LPNT(500)
      DIMENSION XZ(8),YZ(8)
      COMMON  IOPI,TIAC(200),NOAC(200),LIAC(200)
C
C        ***** DESSIN DU RESEAU *****
C
C  --- L'INSTRUCTION  *CALL DESSIN*  PAR LE SUPERVISEUR COMMANDE
C      LE TRACE DU RESEAU SUR IMPRIMANTE A POINTS (VERSATEC).
C
C  --- LE CODE ACTUEL EST TRES PARTICULIER ET RESTREINT AUX CONDITIONS
C      SPECIFIQUES DU SOFTWARE ATTACHE A LA CYBER 74 DE L'U. DE M.
C
C  --- LE DESSIN DES AXES ET DES CENTROIDES A ETE OMIS
C
C**   CALL FACTOR(0.25) POUR VERASTEC
C
C     DESSIN DES NOEUDS ET NUMEROTATION
C
C...  PLUME A L ORIGINE
      CALL PLOT(0.0,0.0,-3)
      DO 602 I=1,NNODE
C...  TRACER UN CERCLE REPRESENTANT LE NOEUD
      CALL SYMBOL(XNO(I),YNO(I),0.04,1,0.0,-1)
C...  ECRIRE LE NUMERO DU NOEUD A 45 DEGRES
      CALL NUMBER(XNO(I)+0.03,YNO(I)+0.03,0.10,NOID(I),45.0,2HI4)
602   CONTINUE
      IF(INDIC.LE.0) GO TO 603
C
C  DESSIN DE L ARBRE D ORIGINE  NOEUD
C
      IF(INDIC.EQ.1) PRINT 101,NOEUD
      IF(INDIC.EQ.2) PRINT 102,NOEUD
  101 FORMAT(5X,*TRAMIN ORIGINE *I5)
  102 FORMAT(5X,*CTM ORIGINE *I5)
      NOEUD=NINT(NOEUD)
      IF(NOEUD.EQ.0) GO TO 630
C           CALCUL DES CHEMINS
      IF(INDIC.EQ.1) CALL TRAMIN(NOEUD)
      IF(INDIC.EQ.2) CALL CATM(NOEUD)
      IF(INDIC.GT.2) GO TO 630
      DROITE=0.0
      DO 75 J=1,NNODE
      IF(XNO(J).GT.DROITE) DROITE=XNO(J)
C...  TRACER UNE ETOILE SUR LE NOEUD ORIGINE
      IF(J.EQ.NOEUD) CALL SYMBOL(XNO(J),YNO(J),0.1,11,0.0,-1)
      LIN=LIAC(J)
```

```
                IF(LIN.EQ.0) GO TO 75
                IO=NOAC(J)
60          C       ON VIENT EN J PAR LA LIGNE LIN DEPUIS LE NOEUD IO
                K=NOLIST(LIN)
             60 NOD=LISTN(K)
                IF(NOD.EQ.J) GO TO 61
                IF(NOD.EQ.IO) GO TO 63
65              K=K+1
                GO TO 60
            C  ON EST SUR LE PARCOURS RETOUR DE LA LIGNE LIN
            C... TRACER UNE FLECHE SUIVIE DU TEMPS DE VOYAGE
            C... A DROITE DU NOEUD J
70          61 CALL SYMBOL(XNO(NOD)+0.03,YNO(NOD),0.1,21,0.0,-1)
                CALL NUMBER(999.0,999.0,0.08,TIAC(NOD),0.0,4HF4.1)
            C... AMENER LA PLUME LEVEE AU NOEUD J
                CALL PLOT(XNO(NOD),YNO(NOD),3)
            C... TRACER LES SEGMENTS DU PARCOURS RETOUR
75          C... DEPUIS LE NOEUD J JUSQU AU NOEUD IO
             62 K=K+1
                IF(NOLR(K).GT.0) GO TO 62
                NOD=LISTN(K)
                CALL PLOT(XNO(NOD),YNO(NOD),2)
80              IF(NOD.EQ.IO) 75,62
            C  ON EST SUR LE PARCOURS ALLER DE LA LIGNE LIN
            C... AMENER LA PLUME LEVEE AU NOEUD IO
             63 CALL PLOT(XNO(NOD),YNO(NOD),3)
            C... TRACER LES SEGMENTS DU PARCOURS ALLER
85          C... DEPUIS LE NOEUD IO JUSQU AU NOEUD J
             64 K=K+1
                IF(NOLR(K).LT.0) GO TO 64
                NOD=LISTN(K)
                CALL PLOT(XNO(NOD),YNO(NOD),2)
90              IF(NOD.EQ.J) 74,64
            C... TRACER UNE FLECHE SUIVIE DU TEMPS DE VOYAGE
            C... A DROITE DU NOEUD J
             74 CALL SYMBOL(XNO(NOD)+0.03,YNO(NOD),0.1,21,0.0,-1)
                CALL NUMBER(999.0,999.0,0.08,TIAC(NOD),0.0,4HF4.1)
95           75 CONTINUE
                DROITE=DROITE+1.5
            C... TRACER UN TITRE A DROITE DU DESSIN
                IF(INDIC.EQ.1) CALL SYMBOL(DROITE,0.5,1.0,6HTRAMIN,90.0,6)
                IF(INDIC.EQ.2) CALL SYMBOL(DROITE,0.5,1.0,3HCTM,90.0,3)
100             GO TO 630
            603 CONTINUE
            C
            C     DESSIN DES TRACES DES LIGNES
            C
105                 DO 620 J=1,NLINE
                   K1=NOLIST(J)
                   K2=NONLIN(J)
                   K=LISTN(K1)
            C... AMENER LA PLUME LEVEE AU NOEUD INITIAL DU PARCOURS
110                CALL PLOT(XNO(K),YNO(K),3)
                   K1=K1+1
                   IFL=0
            C         TRACER LE PARCOURS ALLER
            C         BOUCLE A PARTIR DU DEUXIEME NOEUD
```

```
      DO 610 L=K1,K2
      IF(NOIR(L).EQ.0) GO TO 605
      IFL=1
      IF(NOIR(L).LT.0) GO TO 610
  605 K=LISTN(L)
C...  TRACER LE SEGMENT JUSQU AU PROCHAIN NOEUD ALLER
      CALL PLOT(XNO(K),YNO(K),2)
  610  CONTINUE
      IF(IFL.EQ.1) GO TO 620
C        LA LIGNE EST DIRIGEE, ON SE REPLACE AU NOEUD
C        INITIAL ET ON TRACE LE PARCOURS RETOUR
      K=LISTN(K1-1)
      CALL PLOT(XNO(K),YNO(K),3)
      DO 615 L=K1,K2
      IF(NOIR(L).GT.0) GO TO 615
      K=LISTN(L)
      CALL PLOT(XNO(K),YNO(K),2)
  615 CONTINUE
  620 CONTINUE
C...  RAMENER LA PLUME LEVEE A L ORIGINE
  630  CALL PLOT(0.0,0.0,-3)
C...  FIN DU DESSIN
      CALL ENDPLT
      RETURN
      END
```

2. Tri du fichier ODRES

Comme nous l'avons souligné précédemment, le fichier ODRST (OD-réseau-trié) utilisé par BUILD LINE FLOWS est obtenu en triant ODRES en ordre croissant d'origines. Il a été prévu que ce tri sera effectué en dehors du contrôle du superviseur, de façon à permettre à l'usager l'utilisation du programme de tri du système d'exploitation de son ordinateur local. Cependant, le fichier ODRES ayant une structure particulière, nous avons mis au point un tri spécial (algorithme TRISPE) qui permet de réaliser rapidement le tri en mémoire centrale.

Les caractéristiques de ODRES permettant ce tri spécial sont les suivantes:

- Tous les nombres à trier se situent entre 1 et NBSOM où NBSOM est les nombre de noeuds dans le réseau.

- Plusieurs entrées (O_i, D_j, XP) consécutives de ODRES réfèrent au même O_i , formant ainsi des <u>blocs</u> à l'intérieur du fichier.

Le fonctionnement de TRISPE présenté en [3] pp. 80-88 se résume au principe suivant:

Pour chaque noeud N du réseau, on détermine un chaînage des blocs du fichier dont l'origine est N. Puis, prenant dans l'ordre tous les noeuds du réseau, on regroupe les blocs.

Le tri par destination n'étant par nécessaire pour le module BUILD LINE FLOWS, il a été omis.

Quelques remarques s'imposent avant de présenter TRIODR:

- Le programme a été écrit pour être utilisé sur l'oridinateur CDC CYBER 74 de l'Université de Montréal. Il est possible qu'on doive y effectuer certaines modifications avant de l'utiliser sur une autre installation.

- La longueur du fichier ODRES doit être inférieure ou égale à 24000 entrées. BUILD LINE FLOWS indique cette longueur (nombre de déplacements).

- Le tri sera plus efficace si le nombre de blocs est inférieur à 300.

- Le triplet (O_i , D_j , XP) est compacté en un mot de mémoire. La version actuelle, qui réserve quatre chiffres décimaux pour chaque élément du triplet, nécessite des mots d'au moins 40 bits de longueur. Cependant, si le nombre de noeuds est inférieur à 1000, on pourra utiliser trois chiffres décimaux par élément et réduire cette longueur à 30 bits. On aura alors

$$MUL1 = 1\,000\,000$$

$$MUL2 = 1\,000$$

- L'élément XP du triplet étant un multiple de 0.25, on le transforme en entier en multipliant par 4.0. Sa valeur est rétablie avant la sortie sur ODRST en divisant par 4.0.

- Le fichier ODRES n'est pas détruit par le tri.

- Le fichier ODRST devra contenir, comme premier record, le premier record de ODRES (identificateur).

```
      PROGRAM TRIODR(INPUT,OUTPUT,ODRES,ODRST,TAPE2=ODRES,TAPE3=ODRST)
      COMMON IODR(24000)
      COMMON LIST(9000)
C     TRIODR REALISE LE TRI DE LA MATRICE OD-RESEAU
C     A L AIDE DE L ALGORITHME TRISPE MODIFIE
C     UNE LIMITE DE 24000 ENTREES (IO,ID,XP) EST IMPOSEE
C        ODRES(TAPE2)  EST LE FICHIER A TRIER
C        ODRST(TAPE3)  EST LE FICHIER TRIE EN ORDRE CROISSANT DE  IO
C
C     UNE CARTE DE DONNEES   IDODR,NBSOM   FORMAT(I10,I5)
C        IDODR = IDENTIFICATEUR DU FICHIER ODRES
C        NBSOM = NOMBRE DE SOMMETS DANS LE RESEAU
C
C     N.B.  ODRST AURA LE MEME IDENTIFICATEUR QUE ODRES
C
      MUL1=1000000000
      MUL2=10000
      READ 100,IODDR,NBSOM
  100 FORMAT(I10,I5)
      READ (2) NAME
      IF(EOF(2)) 10,20
   10 PRINT 1000
 1000 FORMAT(*   FICHIER VIDE*)
      STOP
   20 PRINT 200,IODDR,NAME
      IF(IODDR.EQ.NAME) GO TO 30
      PRINT 1001
      STOP
 1001 FORMAT(*  MAUVAIS FICHIER*)
C
C     LECTURE DE TOUTES LES ENTREES DU FICHIER
C
   30 KK=0
   31 READ (2) IO,ID,XD
      IF(EOF(2)) 40,32
   32 KK=KK+1
C
C     NOMBRE DE DEPLACEMENTS TRANSFORME EN ENTIER  (MULTIPLE DE 0.25)
C
      ND=XD*4
      IF(KK.LE.24000) GO TO 35
      PRINT 1002
 1002 FORMAT(*  TROP D ENTREES*)
      STOP
C
C     COMPACTAGE DES TROIS VALEURS EN UN SEUL MOT
C
   35 IODR(KK)=IO*MUL1+ID*MUL2+ND
      GO TO 31
   40 PRINT 202,KK
  202 FORMAT(*  LONGUEUR = *I6)
      REWIND 3
      WRITE (3) NAME
      CALL TRISPE (KK,MUL1,MUL2,NBSOM)
      PRINT 201
  201 FORMAT(*  LE FICHIER EST TRIE*)
      STOP
```

```
      200 FORMAT(*  IDENTIFICATEUR ATTENDU *I10/
     Q* IDENTIFICATEUR TROUVE *I10)
60          END
```

```
      SUBROUTINE TRISPE(L,MULT,MUL2,NBSOM)
      COMMON IRESTE(24000)
      COMMON LDEP(3000),LFIN(3000),LPT(3000)
C
C     EFFECTUE LE TRI D UN FICHIER DE TYPE OD-RESEAU
C
C     LES ENTREES DE OD-RESEAU SONT CONTENUES SOUS FORME COMPACTEE
C     DANS LE VECTEUR  IRESTE
C
C     LE TRI EST EFFECTUE EN ETABLISSANT POUR CHAQUE NOEUD DU RESEAU
C     UN CHAINAGE DES SECTIONS DU VECTEUR IRESTE (BLOCS) AYANT CE NOEUD
C     COMME NOEUD ORIGINE DU DEPLACEMENT,
C     PUIS EN REGROUPANT CES BLOCS DANS L ORDRE CROISSANT DES ORIGINES
C
C     A LA FIN LES ENTREES DE OD-RESEAU TRIEE SONT ENVOYEES SUR TAPE3
C
C     DURANT L EXECUTION, SI LE NOMBRE DE BLOCS ATTEINT LA LIMITE DE 3000
C     LA SOUSPROGRAMME COMPACT EST APPELEE POUR EFFECTUER UN REGROUPEMENT
C     DES BLOCS ET AINSI EN REDUIRE LE NOMBRE
C
C     PARAMETRES
C               - L : NOMBRE D ELEMENTS DE IRESTE A TRIER
C               - MULT : MULTIPLICATEUR POUR COMPACTAGE D L ORIGINE
C               - MUL2 : MULTIPLICATEUR POUR COMPACTAGE DE LA DESTINATION
C               - NBSOM: NOMBRE DE SOMMETS DANS LE RESEAU (<1000)
C
C     COMPACTAGE :  IRESTE(J)= IO*MULT+ID*MUL2+ND
C                   OU  ND=4.0*XJ
C
C     N.B. A L APPEL TAPE3 DOIT CONTENIR L IDENTIFICATEUR
C          ET ETRE POSITIONNE APRES CET IDENTIFICATEUR
C
      IND=L
      LONG=3000
      NCOMP=0
      NSEG=NBSOM
      DO 5 J=1,LONG
      LDEP(J)=0
      LFIN(J)=0
    5 LPT(J)=0
      J=1
      IPREC=IRESTE(J)/MULT
      K=IPREC
      LDEP(K)=J
   10 J=J+1
      IF(J.GT.IND) GO TO 100
      IF(IRESTE(J)/MULT - IPREC) 20,10,20
   20 K=IPREC
   30 IF(LFIN(K).NE.0) GO TO 40
      LFIN(K)=J-1
      GO TO 50
   40 K=LPT(K)
      GO TO 30
   50 K=IRESTE(J)/MULT
      IF(LDEP(K).NE.0) GO TO 60
      LDEP(K)=J
      GO TO 80
```

```
   60 IF(LPT(K).NE.0) GO TO 70
      NSEG=NSEG+1
      IF(NSEG.LE.LONG) GO TO 65
C
C     REDUIRE LE NOMBRE DE BLOCS
C
      CALL COMPACT(NBSOM,NSEG)
      NCOMP=NCOMP+1
      GO TO 50
   65 LPT(K)=NSEG
      LDEB(NSEG)=J
      GO TO 80
   70 K=LPT(K)
      GO TO 60
   80 IPREC=IRESTE(J)/MULT
      GO TO 10
  100 K=IPREC
  105 IF(LFIN(K).NE.J) GO TO 110
      LFIN(K)=J-1
      GO TO 120
  110 K=LPT(K)
      GO TO 105
  120 DO 200 I=1,NBSOM
      IF(LDEB(I).EQ.0) GO TO 200
      II=I
  125 I1=LDEB(II)
      I2=LFIN(II)
      DO 150 J=I1,I2
      MOT=IRESTE(J)
      MOT=MOT-1*MULT
      ID=MOT/MUL2
      ND=MOT-ID*MUL2
      XD=ND/4.0
      WRITE (3) I,ID,XD
  150 CONTINUE
      IF(LPT(II).EQ.0) GO TO 160
      II=LPT(II)
      GO TO 125
  160 CONTINUE
  200 CONTINUE
      PRINT 222,NCOMP
  222 FORMAT(///,5X,I3,* COMPACTAGES*/)
      RETURN
      END
```

```
      SUBROUTINE COMPACT(NBSOM,NSEG)
      COMMON IODR(24000)
      COMMON LDEP(3000),LFIN(3000),LPT(3000)
C
C     EFFECTUE UN TRI PARTIEL POUR DIMINUER LE NOMBRE DE BLOCS (NSEG)
C     ET RETABLIT LES CHAINAGES
C
C     TAPE3 (ODRST) EST UTILISE COMME FICHIER DE TRAVAIL
C     MAIS SON IDENTIFICATEUR EST PROTEGE
C     ET LE FICHIER EST REPOSITIONNE APRES CET IDENTIFICATEUR
C     A LA FIN DU COMPACTAGE
C
C
C     LA BOUCLE 200  -REGROUPE LES BLOCS EN ORDRE CROISSANT D ORIGINE
C                    -LES ECRIT SUR TAPE3
C                    -REPLACE LES POINTEURS DES LISTES
C
      K=0
      DO 200 I=1,NBSOM
      IF(LDEP(I).EQ.0) GO TO 200
      II=I
      LDP=K+1
  125 I1=LDEP(II)
      I2=LFIN(II)
      DO 150 J=I1,I2
      K=K+1
      WRITE (3) IODR(J)
  150 CONTINUE
      IF(LPT(II).EQ.0) GO TO 160
      II=LPT(II)
      GO TO 125
  160 LDEP(I)=LDP
      LFIN(I)=K
      LPT(I)=0
  200 CONTINUE
C
C     MISE A ZERO DES POINTEURS DEVENUS INUTILES
C
      I1=NBSOM+1
      I2=NSEG-1
      DO 300 I=I1,I2
      LDEP(I)=0
      LFIN(I)=0
      LPT(I)=0
  300 CONTINUE
C
C     LES ENTREES ORDONNEES ET BLOQUEES SONT REPLACEES DANS LE VECTEUR
C
      REWIND 3
      READ (3) ID
      DO 350 J=1,K
      READ (3) IODR(J)
  350 CONTINUE
      REWIND 3
      WRITE (3) ID
      NSEG=NBSOM
```

```
      RETURN
      END
```

REFERENCES

[1] ACHIM, C., CHAPLEAU, C., CHRIQUI, C. et ROBILLARD, P., "TRANSCOM - Guide de l'utilisateur", publication no 23, Centre de recherche sur les transports, Université de Montréal, juin 1975.

[2] CHAPLEAU, R., "Réseaux de transport en commun: structure informatique et affectation", publication no 13, Centre de recherche sur les transports, Université de Montréal, juin 1974.

[3] CHRIQUI, C., "Réseaux de transports en commun: les problèmes de cheminement et d'accès", publication no 11, Centre de recherche sur les transports, Université de Montréal, avril 1974.